NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

QUADROTOR INTERCEPT TRAJECTORY PLANNING AND SIMULATION

by

Robert L. Allen III

June 2017

Thesis Advisor:	Xiaoping Yun
Co-Advisor:	Marcello Romano
Second Reader:	Robert Hutchins

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704–0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2017	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
QUADROTOR INTERCEPT TRAJECTORY PLANNING AND SIMULATION

5. FUNDING NUMBERS

6. AUTHOR(S) Robert L. Allen III

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

Quadrotor drones pose a safety hazard when operated in or near controlled airspace. A hazardous quadrotor could be intercepted and removed by another quadrotor. In this thesis, we seek to determine if optimal control methods outperform missile control methods when applied to a quadrotor drone performing an intercept with a moving target. This is achieved by simulating the intercept of a target with a quadrotor and comparing the performance of several on-line trajectory planners. Two missile control-based trajectory planners, pursuit guidance and proportional navigation, are compared against an optimal control trajectory planner. The time and energy used by a simulated quadrotor to intercept a target are the performance measures used for comparison. The trajectory planners use a three-degree of freedom model, and the simulated quadrotor uses a six-degree of freedom model. Each trajectory planner is compared in a crossing, head-on, and tail-chase geometry. All of the on-line results are compared to an off-line optimal solution. The results show that the off-line optimal control method performs better than the on-line trajectory planners, regardless of intercept geometry type. The proportional navigation planner has the best performance of the on-line trajectory planners.

14. SUBJECT TERMS quadrotor drone, micro interceptor, trajectory planner, optimal control, missile guidance law, intercept trajectory, optimal trajectory	**15. NUMBER OF PAGES** 91
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540–01-280-5500 Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

QUADROTOR INTERCEPT TRAJECTORY PLANNING AND SIMULATION

Robert L. Allen III
Lieutenant, United States Navy
B.S., Virginia Military Institute, 2010

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2017

Approved by: Xiaoping Yun
Thesis Advisor

Marcello Romano
Co-Advisor

Robert Hutchins
Second Reader

R. Clark Robertson
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Quadrotor drones pose a safety hazard when operated in or near controlled airspace. A hazardous quadrotor could be intercepted and removed by another quadrotor. In this thesis, we seek to determine if optimal control methods outperform missile control methods when applied to a quadrotor drone performing an intercept with a moving target. This is achieved by simulating the intercept of a target with a quadrotor and comparing the performance of several on-line trajectory planners. Two missile control-based trajectory planners, pursuit guidance and proportional navigation, are compared against an optimal control trajectory planner. The time and energy used by a simulated quadrotor to intercept a target are the performance measures used for comparison. The trajectory planners use a three-degree of freedom model, and the simulated quadrotor uses a six-degree of freedom model. Each trajectory planner is compared in a crossing, head-on, and tail-chase geometry. All of the on-line results are compared to an off-line optimal solution. The results show that the off-line optimal control method performs better than the on-line trajectory planners, regardless of intercept geometry type. The proportional navigation planner has the best performance of the on-line trajectory planners.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

3D	three-dimensional
3-DOF	three-degree of freedom
6-DOF	six-degree of freedom
CAD	computer aided design
CPA	closest point of approach
DC	direct current
ESC	electronic speed control
FAA	Federal Aviation Administration
GNC	guidance navigation and control
GPS	Global Positioning System
LQR	linear quadratic regulator
LQT	linear quadratic tracker
MPC	model predictive control
NED	north east down
PG	pursuit guidance
PID	proportional-integral-derivative
PN	proportional navigation
UAV	unmanned aerial vehicle
VR	virtual reality

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

Thank you to

Dr. Yun and Dr. Romano for being patient and advising me on this research,

Dr. Hutchins for teaching me the basics of missile guidance,

Dr. Kaminer for expanding my understanding of optimal control,

Dr. Park and the Spacecraft Robotics Lab for the support I needed to get started writing a simulation, and

Dr. Jones for your support measuring motor performance and providing expert knowledge of quadrotors.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Unmanned aerial systems are increasingly common in both military and civilian environments. With the advent of inexpensive direct current (DC) motors and motor controllers, quadrotor unmanned aerial vehicles (UAVs) have become widely available in the civilian hobby sector. Systems as simple as line-of-sight pilot-controlled model aircraft and as complex as autonomous Global Positioning System (GPS) guided camera systems now crowd the already saturated airspace. As these unmanned craft have become more available, regulating their use has become more challenging.

The Federal Aviation Administration (FAA), the agency responsible for the safe use of airspace nationwide, has taken several steps to educate the public on the proper use of unmanned aircraft. This approach to safety has its limits and best serves amateur operators who have sought out information on regulations and safety restrictions. Unlike a licensed pilot who is required to submit to regular oversight by the FAA when operating manned aircraft, no system ensures that a private citizen who builds or purchases a remotely operated drone abides by the applicable airspace regulations. A gap in enforcement capabilities remains in locations where safety of flight is critical—namely in and around controlled airspace at airports.

Several commercially available systems can assist in the enforcement of regulations for remotely operated hobby quadrotor UAVs. Most commonly, systems like Skytracker [1] use a series of sensors to triangulate the rogue UAV based on signals emitted by the quadrotor and its ground controller. The triangulated ground position is then forwarded to authorities who are left with the burden of getting the pilot to land the system safely. In some cases, the detection systems may also be equipped with a jammer that can disable a system and cause it to enter a fail-safe hover mode.

More aggressive systems are being developed to remove rogue UAVs from controlled airspace. Some systems use lasers or small projectiles to immediately remove the safety threat. Although firing a projectile may effectively remove the UAV, it can

create a hazard, especially if the projectile might land in a populated area. A different method is needed to remove a rogue UAV in an area where shooting it down is unsafe.

Arguably, the safest method of removing a rogue UAV is to catch it. This also allows the operator of a registered but rogue UAV to be identified so that a fine or other penalty may be assessed. Companies such as Guard From Above use trained raptors that grab the rogue UAV and return with it in hand. Once captured, the UAV is no longer a safety threat and does not fall onto people or property [2]. This approach has its own risks and advantages, considering the safety of the bird or the possibility that the bird may drop the UAV.

If we can design a system for airports that detects small rogue aircraft by using a radar or camera, then we can use a quadrotor UAV to remove the rogue safety threat. A central control system can be used to send a “micro interceptor” to intercept the rogue UAV and neutralize it by capture or destructive force, as appropriate. The purpose of this research is to compare different methods of on-line trajectory planning for a quadrotor micro interceptor used in this capacity to intercept a moving target.

A. BACKGROUND

The intercept problem has analogues in several fields. Missile guidance is directly related because it shares the same goal: to intercept a remotely detected target on the command of a human operated system and prevent the target from potentially causing harm to a protected area or object.

An active missile is guided to a target based on energy emitted by the missile that reflects off the target, usually a radar. A semi-active missile is guided to a target based on energy emitted by a separate station that reflects off the target, such as a transmitter located at the launcher. A passive missile is guided to a target based on measuring energy that is emitted by the target, such as heat or radar signals. Hybrids of these methods are often used for guidance during phases of flight that may not allow for useful transfer of information. The type of guidance a missile uses is determined by the type of platform that fires the missile and the threat the missile is expected to encounter.

In this research, the hypothetical system is assumed to have perfect knowledge of the target and the micro interceptor. Because payload weight limits on the interceptor preclude the use of robust onboard sensors for tracking the target, the burden of trajectory planning is assumed by the ground control station. This allows the control of the interceptor to be limited to a command position. The sequence in time of command positions composes the command trajectory. The interceptor compares its known position to the command position and internally produce the required flight control inputs to fly the commanded trajectory. Unlike most missiles, this system receives a trajectory to fly from the ground controller.

A missile system operates in a highly time-constrained environment because often the threats to such a system move quickly and in ways that are intentionally difficult to detect. This requires that the missile also be moving quickly, often pushing the limits of forces applied to the airframe. Similarly, this research examines the limits of the micro interceptor with respect to power consumption and time performance, assuming a threat with comparable flight characteristics.

B. THESIS OUTLINE AND OBJECTIVES

In this research, we seek to determine if optimal control can outperform classic missile control methods when applied to quadrotor drones by simulating the intercept of a target with a quadrotor drone and comparing the performance measure of three trajectory planners. The trajectory planner with the best performance in simulation represents the best planner for actual flight trajectory planning.

Applicable background in guidance, navigation, and control (GNC) and a brief outline of classic missile guidance are given in Chapter II. The theoretical background of optimal control and how related works have solved similar intercept trajectory planning problems is also discussed.

Next, the dynamic model for a simulated quadrotor is presented in Chapter III. The assumptions made in simplifying the dynamic model that allow simulation using Simulink and MATLAB are stated and explained.

The trajectory planners used by the simulation are explained in Chapter IV. The point mass model and the guidance law used for each planner to create a trajectory are discussed.

The Simulink and MATLAB simulation environment is explained in Chapter V. The function of every piece of the time-based simulation is outlined and technical parameters are defined.

The framework of the simulated experiment is described in Chapter VI. Three intercept geometries are defined, and the method of evaluating a trajectory planner is established. The results of the simulation can be used to make a conclusion based on the evaluation of the trajectory planner.

The results of the simulation are presented in Chapter VII. Results are shown as a collection of trajectory plots and flight statistics, grouped by intercept geometry. The flight statistics provide information about the experiment that are not observable in a trajectory plot (velocity and distance to target) and highlight the strengths and weaknesses of each planner against the different geometries.

Finally, a conclusion based on the simulated data states which trajectory planner has the highest performance in Chapter VIII. A summary of shortcomings of the research as well as areas for future work are presented.

II. LITERATURE REVIEW AND RELATED WORKS

GNC refers to the process of solving the intercept problem according to Lin in [3]. Lin explains that typically, this problem involves an intercept vehicle and a target, although it may also refer to other problems in aerospace or robotics. Lin states that navigation refers to the system function on the intercept vehicle that determines its attitude and position, guidance refers to the system function that determines a trajectory (physical flight path) to produce an intercept with the target, and control refers to the function of translating guidance commands to flight surface and thrust inputs. Control ensures that the vehicle is stable in flight while following guidance commands and rejecting system disturbance [3].

An example of a GNC system is a missile system, such as those described in [4] and [5]. In these systems, a missile is the intercept vehicle, and the target is another missile or an aircraft. The missile may use an inertial navigation system or an external tracking system (such as a radar) to determine its position. This raw measurement of position and attitude is filtered with a Kalman filter (or similar) to produce a quality estimate of the actual position and attitude of the missile. The navigation information is compared to the position information of the target by the guidance computer. The guidance computer produces a command trajectory that produces an acceptable intercept. The intercept is determined based on an acceptable intercept distance, which is determined by the range required to deploy the payload (explosive charge in the case of missiles).

Missile systems solve the air-to-air intercept problem effectively and serve as the foundation for solving the same problem applied to quadrotor UAVs. Methods for quadrotor intercept have been explored in literature. Pursuit guidance, a missile guidance law, is adapted by [6] for the purposes of quadrotor path following in confined spaces. Further, proportional navigation (PN) is adapted for ground target intercepts by [7]. The ground based target is limited to two dimensions for maneuvering, and the quadrotor navigates in two dimensions while maintaining a fixed altitude. A similar application of

PN is explored in [8], which tracks a flying target although in this case the target is free to maneuver in all three dimensions, the problem remains constrained in altitude.

In addition to adapting missile guidance law to generate intercept trajectories, researchers have applied the principles of optimal control to intercept trajectory planning. A time-optimal trajectory planner is developed and applied to a quadrotor in [9] and [10], where the goal is to move to a series of fixed waypoints. Similarly, optimal trajectories for moving target intercept in real time are developed by [11] and references therein. A method for testing trajectory planners is explored in [12], following different trajectory geometries that may be suited for specific types of missions. Further integrated mission planning using time optimal guidance and optimal control are developed by [13] and executed in hardware.

Flight control of a quadrotor is also widely studied. A common method for producing flight commands to follow a trajectory is a proportional-integral-derivative (PID) controller. A PID controller seeks to minimize the error between a reference input and measured state. PID controllers and analysis methods outlined by [14] can be applied to the flight control of a quadrotor. These methods are demonstrated by [15] to control the thrust produced by the motors in order to execute guidance law. Additionally, [16] shows a similar approach but instead uses an optimal linear quadratic regulator (LQR) method for flight control.

Optimal control methods differ from classic control methods because they seek to minimize a cost function. The cost function and control input are related by a function called a Hamiltonian, which is a function of system states and co-states [17].

Literature widely documents the challenges associated with creating an on-line intercept (an intercept in real time). The challenge stems from knowing only where the target was and is without any knowledge of where the target is going or how it will arrive there. The solutions to the on-line intercept problem are sub-optimal, meaning that the cost to perform an intercept will not be the theoretical minimum.

The pursuit guidance and proportional navigation methods for missile guidance given by [4] and [5] are on-line solutions to the intercept problem. As shown in [7], those

guidance methods are at least suitable for a quadrotor aircraft performing an intercept with a ground target. Other attempts, such as in [6], demonstrate that proportional navigation is a feasible method for a three-dimensional (3D) intercept of an airborne target.

THIS PAGE INTENTIONALLY LEFT BLANK

III. DYNAMIC MODELS

A model is used to represent the command trajectories as well as the simulated quadrotor. A point mass model with only turn rate as an input represents the command trajectories created by pursuit guidance and proportional navigation planners. A point mass model with a three-dimensional acceleration input is used to represent the command trajectory for the model predictive control (MPC) planner. The point mass model used to represent the simulated quadrotor is controlled by a thrust (force) and torque input. In this section, we show the equations of motion and define all reference frames for each one of the point mass models.

A. COORDINATE AND AXIS DEFINITION

The coordinate system used for the inertial frame is north east down (NED), as shown in Figure 1.

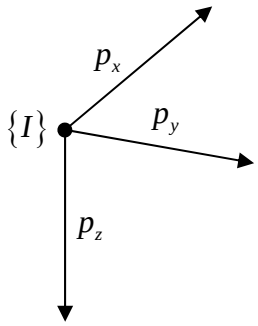


Figure 1. Inertial Coordinates

The coordinates of the target and the interceptor are expressed as a vector originating from the inertial frame

$$p = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}. \tag{1}$$

The trajectory for both target and interceptor is expressed as a discrete position with coordinates p. Trajectories are generated using a point mass model represented by three degrees of freedom.

B. MODELING ASSUMPTIONS

Aerodynamics are not modeled for either the quadrotor model or the trajectory model. The state of the quadrotor battery is assumed to remain constant for duration of flight, providing a constant voltage and current. This allows the motor performance to be modeled as a constant, neglecting any change in thrust-to-throttle caused by fluctuation in battery voltage. Finally, the position and attitude of the quadrotor and the target are assumed to be known at all times.

C. EQUATIONS OF MOTION

Three-degree of freedom (3-DOF) and six-degree of freedom (6-DOF) models are defined in this section. The models are used to generate trajectories and simulate the flight of a quadrotor. They are implemented using MATLAB S-functions.

1. Trajectory 3-DOF Model

Two models are used to create the command trajectory. The first model is for the pursuit guidance and proportional navigation planners, adapted from [18]. It consists of six states

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \\ \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -\omega & 0 \\ 0 & 0 & 0 & \omega & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \end{bmatrix} \tag{2}$$

with

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \end{bmatrix}. \tag{3}$$

The vector p is the position of the trajectory defined in Figure 1. The velocity is the vector v, and ω is the commanded turn rate (control input). This nonlinear model uses a fixed velocity, and the control input is described in a later section.

The model predictive controller does not produce a commanded turn rate. Instead, it controls a three-dimensional force vector. This allows a linear model of the standard form $\dot{x} = Ax + Bu$ to be used such that

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \\ \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{A} \begin{bmatrix} p_x \\ p_y \\ p_z \\ \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{B} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}. \tag{4}$$

The control input in (4) is the force. In this case, force is defined by $f = m\bar{a}$ with mass $m = 1$. This allows the input to be simplified to

$$u = \bar{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}. \tag{5}$$

The guidance law governing the control input for this planner is described in a later section.

2. Aircraft 6-DOF Model

Equations of motion representing the simulated quadrotor are expressed as a 6-DOF point mass. This model is adapted from [19] and is derived using the Newton-Euler approach. The state vector for the simulated quadrotor is given as

$$\underbrace{\begin{bmatrix} p \\ v \\ \Phi \\ \omega \end{bmatrix}}_{12\times 1} \tag{6}$$

where

$$p = \underbrace{\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}}_{3\times 1}, \tag{7}$$

$$v = \underbrace{\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}}_{3\times 1}, \tag{8}$$

$$\Phi = \underbrace{\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}}_{3\times 1}, \tag{9}$$

and

$$\omega = \underbrace{\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}}_{3\times 1}. \tag{10}$$

In Equations (6) through (10), the vector p is the position in the inertial frame, the vector v is the velocity in the inertial frame, the vector Φ is the orientation expressed as Euler angles, and ω is the angular rate. The Euler angles are defined in 0Roll is represented by the angle ϕ, pitch is represented by the angle θ, and the angle ψ is yaw.

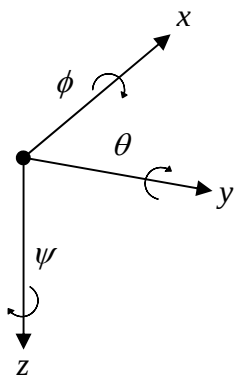


Figure 2. Roll, Pitch, and Yaw Angles

The time derivatives of Equation (6) are given by

$$\dot{p} = v\,, \tag{11}$$

$$\dot{v} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} - R_{ib} \begin{bmatrix} 0 \\ 0 \\ f_t / m \end{bmatrix}, \tag{12}$$

$$\dot{\Phi} = Q^{-1}\omega\,, \tag{13}$$

and

$$\dot{\omega} = J^{-1}\left(\tau - skew(\omega)J\omega\right). \tag{14}$$

The force due to gravity is g, f_t is the force due to thrust, m is the mass of the quadrotor, and τ is the total torque produced by the four motors. The system is a function of the control inputs f_t and τ illustrated in Figure 3.

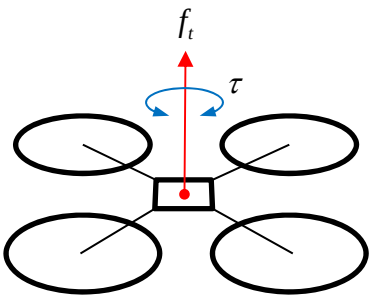


Figure 3. Control Inputs

Additionally, the components of (13) and (14) are defined by

$$Q = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix}, \tag{15}$$

$$J = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix}, \tag{16}$$

and

$$skew(\omega) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{17}$$

The components J_x, J_y, and J_z are the inertial moments along each axis. The rotation matrices are defined as for each axis as

$$R_\psi = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{18}$$

$$R_\theta = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \tag{19}$$

and

$$R_{\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}. \tag{20}$$

Equations (18) to (20) are combined, producing a single rotation matrix

$$R_{ib} = R_{\psi} R_{\theta} R_{\phi} . \tag{21}$$

Orientation of the quadrotor is expressed in the inertial frame as an orientation defined by Euler angles using R_{ib} .

THIS PAGE INTENTIONALLY LEFT BLANK

IV. INTERCEPT TRAJECTORY PLANNER TYPES

There are three trajectory planners used in this research. Pursuit guidance (PG) and PN are commonly used for missile trajectories. MPC is widely applied in the field of controls and is adapted to trajectory planning. Each of these methods uses the current position of the interceptor and the target in order to calculate an intercept trajectory.

The linear quadratic tracker (LQT) is an optimal control method used to track a reference trajectory. This trajectory planner is offline (not real time) and serves as a benchmark to compare the performance of the three on-line trajectory planners. In this section, we describe all four of these trajectory planning methods.

A. PURSUIT GUIDANCE

PG is the simplest of the trajectory planners. The method is adapted from missile guidance law that produces an intercept with a target by pointing the heading of the missile at the target for the duration of flight. To achieve this, the only input the algorithm needs is the angle between the missile and the target. This is used to generate a turn rate ω that controls the heading of the missile.

To adapt this method for the purpose of an intercept between two quadrotors, a point mass model is used to represent the interceptor to produce an on-line command trajectory. The point mass model is manipulated with a single input that controls the turn rate, constrained to two dimensions in the x-y plane (a yawing turn along the z-axis).

This control excludes the z-altitude axis, which is assumed to be a relatively steady parameter in a quadrotor target. The amount of motion in this axis is likely to be significantly lower than in the x-y plane because many commercially available quadrotors are controlled with the altitude fixed to minimize the complexity of flight control for the pilot.

From [18], the turn rate is determined by

$$\omega = k\left(\tan^{-1}\left(\frac{P_y}{P_x} \right) - \tan^{-1}\left(\frac{v_y}{v_x} \right) \right). \tag{22}$$

The gain k can be tuned to produce more or less aggressive turning maneuvers. In the case of a missile in flight, this may be an important consideration given that the forces acting on the missile may exceed the design rating. In the case of a quadrotor, however, the force design margins are not as tight. The line of sight vector P is defined by

$$P = \begin{bmatrix} x_t - x_i \\ y_t - y_i \end{bmatrix} \tag{23}$$

which is the vector formed by the position difference of the interceptor and the target. Vectors used in the pursuit guidance planner and Equation (23) are illustrated in Figure 4.

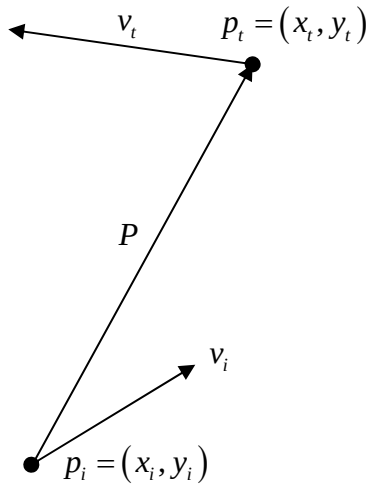


Figure 4. Vector Definitions for Pursuit Guidance

This trajectory planner assumes a fixed velocity. The starting position and velocity are given as initial conditions in the S-function used to implement the differential equation, and velocity is selected to be two meters per second faster than that of the target. This ensures that the interceptor will catch the target in a worst-case

scenario tail-chase geometry. For physical implementation, the velocity should be selected to be slightly below the known maximum flight speed of the interceptor.

B. PROPORTIONAL NAVIGATION

PN is an improvement on the PG planner. Instead of pointing the heading of the trajectory directly at the target (as with PG), PN steers the trajectory on a course that causes an intercept to occur in the path of a non-maneuvering target. The resulting trajectory intercepts a target by maintaining a constant line-of-sight angle. To achieve this, the additional input of closing velocity must be provided to calculate turn rate.

To adapt this method for the purpose of an intercept between two quadrotors, a point mass model is used to represent the interceptor to produce an on-line command trajectory. The point mass model is manipulated with a single input that controls the turn rate, constrained to two dimensions in the x-y plane as previously described with the pursuit guidance planner. The same 3-DOF model is used with the input ω given in [18] as

$$\omega = \frac{k v_c \dot{\theta}_{LOS}}{v_i \cos(\theta_i - \theta_{LOS})} \tag{24}$$

where k is a gain, v_c is the closing velocity between interceptor and target, v_i is the velocity of the interceptor, and θ_{LOS} is the line-of-sight angle between v_c and the y-axis.

The vectors used for PN are illustrated in Figure 5. The relative velocity v_r between target and interceptor is defined for reference. As with the PG planner, the PN planner is initialized with a starting position and velocity that remains constant throughout the flight.

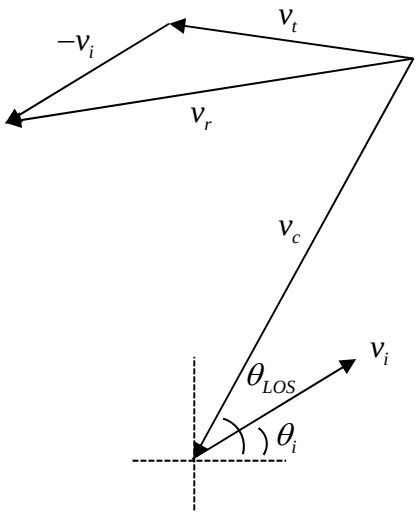


Figure 5. Vector Definitions for Proportional Navigation

C. OPTIMAL TRAJECTORIES

The off-line optimal trajectory planner is adapted from [17]. The LQR method solves the optimal trajectory problem given a system of the form

$$\dot{x} = Ax + Bu \tag{25}$$

where x is the state vector, $\dot{x}$ is the time derivative of the state vector, and u is the input vector with A and B as constant matrices. The system is subject to the cost

$$J = \frac{1}{2}x^T(t_f)Hx(t_f) + \frac{1}{2}\int_{t_0}^{t_f}\left[x^T(t)Qx(t) + u^T(t)Ru(t)\right]dt \tag{26}$$

with H, Q, and R as real and positive definite matrices. From the cost, the Hamiltonian is formed

$$H(x,u,p,t) = \frac{1}{2}\left[x^T(t)Qx(t) + u^T(t)Ru(t)\right] + p^T\left[Ax + Bu\right] \tag{27}$$

where p is the costate vector. The necessary conditions for optimality are given by

$$\dot{x} = \frac{\partial H}{\partial p} = Ax + Bu\,, \tag{28}$$

$$\dot{p} = -\frac{\partial H}{\partial x} = -Qx - A^T p\,, \tag{29}$$

and

$$0 = \frac{\partial H}{\partial u} = Ru + B^T p\,. \tag{30}$$

Finally, the optimal input $u^*(\mathrm{t})$ for an open loop system is given by

$$u^*(t) = -R^{-1}B^T p(t)\,. \tag{31}$$

A more useful solution is the LQT method. Instead of finding an optimal control that drives the state to zero, the LQT finds an optimal control that drives the state to a time varying reference trajectory. In this case, the cost J is given by

$$J = \frac{1}{2}\left[x(t_f) - r(t_f)\right]^T H\left[x(t_f) - r(t_f)\right] + \frac{1}{2}\int_{t_0}^{t_f}\left[\left[x(t) - r(t)\right]^T Q\left[x(t) - r(t)\right] + u^T(t)Ru(t)\right]dt \tag{32}$$

where r is the reference trajectory. The Hamiltonian is given by

$$H(x,u,p,t) = \frac{1}{2}\left[\left[x(t) - r(t)\right]^T Q\left[x(t) - r(t)\right] + u^T(t)Ru(t)\right] + p^T\left[Ax + Bu\right]. \tag{33}$$

The necessary conditions of optimality become

$$\dot{x} = \frac{\partial H}{\partial p} = Ax + Bu\,, \tag{34}$$

$$\dot{p} = -\frac{\partial H}{\partial x} = -Q(x - r) - A^T p\,, \tag{35}$$

and

$$0 = \frac{\partial H}{\partial u} = Ru + B^T p\,. \tag{36}$$

Finally, the optimal control becomes

$$u^*(t) = -R^{-1}B^T p(t)\,. \tag{37}$$

This method is used to generate an optimal trajectory to serve as a comparison to the on-line solutions created in the simulation.

D. MODEL PREDICTIVE CONTROL

MPC is described in [20] and [21]. MPC, like optimal control, produces a control output for a system that minimizes a cost function. Unlike LQT, MPC uses a model of the system to predict the output and internal state at given time in the future. The forward looking time period is called the horizon. MPC calculates an optimized control input at every time step considering the predicted output at the time horizon. This behavior is desirable for a wide range of applications but may be particularly well suited for intercept trajectory planning.

In addition to predicting the output and state of a controlled system, MPC may also be used to constrain internal states of the system. This is applied to the intercept problem to limit the velocity of the command trajectory. Limiting the force (system input) applied to the trajectory planner will not prevent the velocity of the command trajectory from exceeding the capabilities of the quadrotor. Instead, a constraint is created on the velocity state in the MPC's internal model of the system.

The controller used in this research is included in the MATLAB and Simulink controls library as the MPC toolbox. The toolbox documentation [22] describes the cost function optimized by the MPC controller. The cost function is similar to the function given in Equation (32) but includes additional terms that account for algebraic constraints. Before the block can be used in Simulink, an MPC object must be present in the workspace. This is achieved before loading the simulation using a MATLAB script. The MPC object is initialized with the system model in the form $\dot{x} = Ax + Bu$ and $y = Cx + Du$, specifically,

$$\dot{x} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} u \tag{38}$$

and

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} u . \tag{39}$$

System state constraints are specified to limit the velocity in the form of $Eu + Fy \le G$. This is given by

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} u + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} y \le \begin{bmatrix} 0 \\ 0 \\ 0 \\ v_{\max} \\ v_{\max} \\ v_{\max} \end{bmatrix} \tag{40}$$

where $v_{\max}$ is the maximum velocity in each direction. For this simulation, $v_{\max}$ is chosen to be a soft constraint of nine meters per second. The initialization function also sets the time step $T_s = 0.1$ s, prediction horizon $p = 50$, and the control horizon $m = 20$.

The MPC trajectory planner is shown in Figure 6. The Simulink MPC block inputs are the reference (ref) signal and measured outputs (mo). The MPC block outputs are manipulated variables (mv). For the MPC trajectory planner, the ref signal is the target trajectory, the mo signal is the current position of the trajectory, and the mv signal is the force input to the S-function for trajectory dynamics.

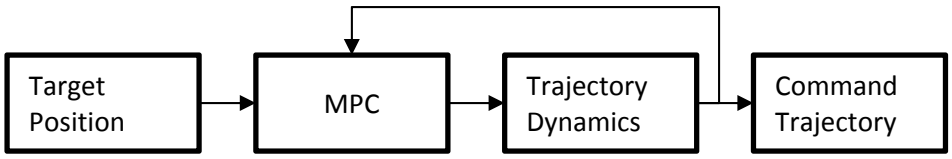


Figure 6. MPC Trajectory Planner Description

V. SIMULINK IMPLEMENTATION

To examine the performance of the trajectory planners, a simulated experimental environment was created using MATLAB and Simulink. The environment uses the equations of motion previously described to simulate the flight of the target, the intercept trajectory, and the flight of the quadrotor. This is achieved using a combination of user defined MATLAB functions and S-functions. The Simulink models are described in this section, and the supporting MATLAB scripts and functions are provided in Appendix B.

A. OVERVIEW

Each simulated flight starts the target, command trajectory, and quadrotor at a specified starting point with an initial velocity. These parameters are defined as a part of geometry selection. Additionally, each simulated flight uses one trajectory planner. The trajectory planner and geometry are selected before the simulation begins.

A block diagram showing the relationship of each component of the simulation is shown in Figure 7. The target dynamics, trajectory planner, and quadrotor dynamics are differential equations implemented with an S-function. The flight controller is implemented as a MATLAB function. The 3D visualizer uses blocks from the virtual reality toolbox to create a virtual environment to observe the position and attitude of the quadrotor throughout the intercept. Trajectory plots and flight data are the experimental results.

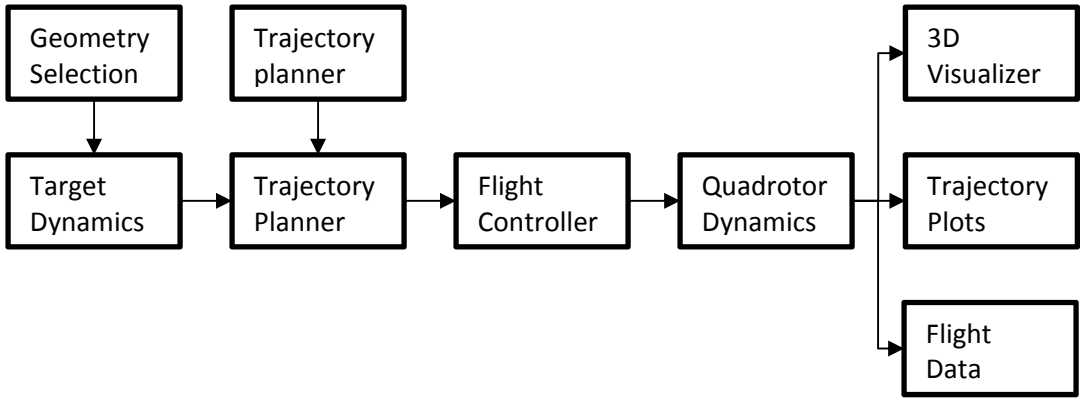


Figure 7. Structure of Simulation

B. TRAJECTORY PLANNERS

The trajectory planners are illustrated in Figure 8. Target position and velocity are inputs and the command trajectory is the output. The guidance law block is implemented using a MATLAB function, and the interceptor dynamics block is implemented with an S-function.

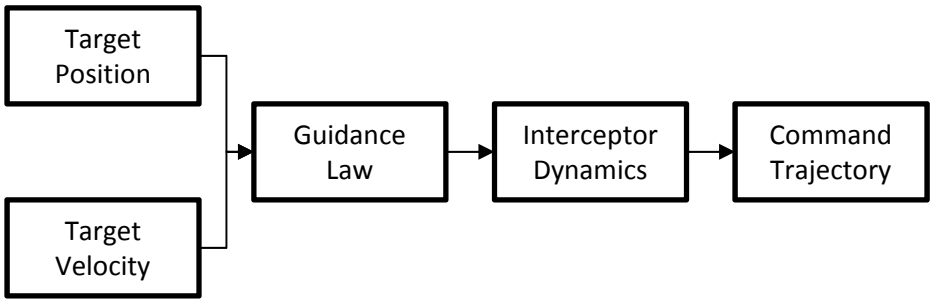


Figure 8. Trajectory Planner Block Diagram

C. QUADROTOR FLIGHT CONTROL

The purpose of the flight controller is to produce thrust and torque commands to fly the quadrotor to the command trajectory. Because the effects of aerodynamic drag are not modeled, the flight controller must impose a maximum velocity that is realistic of a quadrotor experiencing those effects. To do this, two PID controllers are used to isolate velocity as an intermediate variable which can be manipulated.

The configuration of the PID controllers, adapted from [15], is shown in Figure 9. Each controller is implemented with a Simulink PID block. Position, velocity, gravity, and force are all represented by three-dimensional vectors.

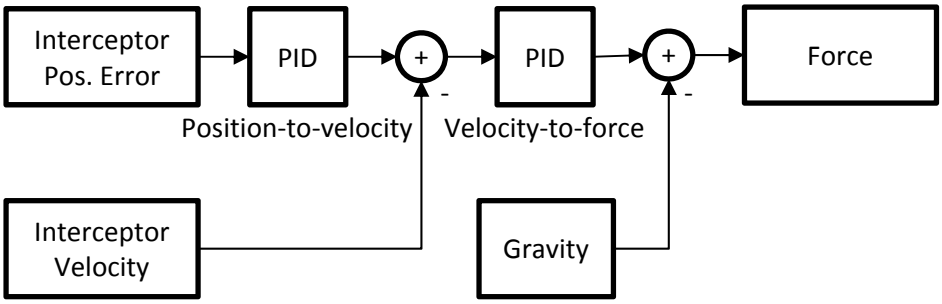


Figure 9. PID Configuration in Flight Controller

The configuration of each PID block is shown in Table 1. Saturation is used to control the maximum velocity that the quadrotor flies as it moves to the command position.

Table 1. Simulink PID Block Parameters

Block Parameter	**Position-to-velocity**	**Velocity-to-force**
Proportional (P)	1.3	1.5
Integral (I)	0.1	0.01
Derivative (D)	0.7	0.7
Filter Coefficient (N)	100	100
Saturation Limit	±20	±10

The command thrust f_t and command torque τ are produced based on input force, quadrotor orientation Φ, and angular rate ω as shown in Figure 10. The flight commands block is a MATLAB function that creates command thrust and a command orientation Φ_{out}. The orientation error is produced by subtracting quadrotor orientation from command orientation. A PID block is used to produce command torque based on orientation error.

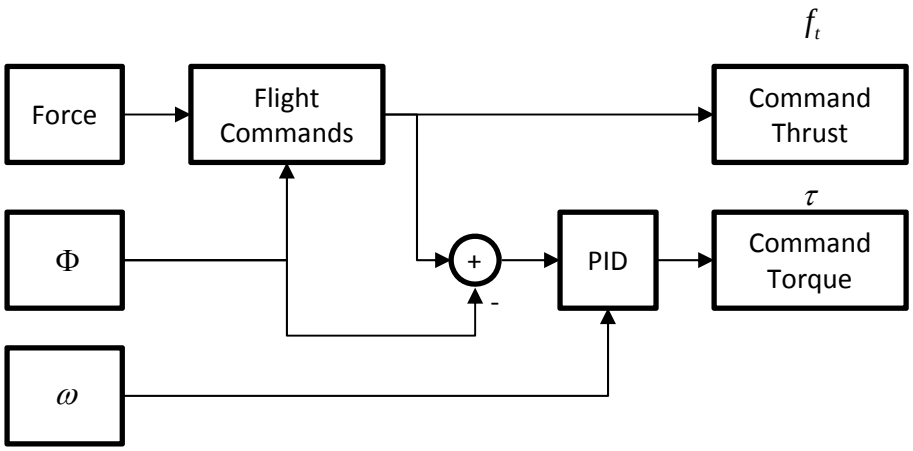


Figure 10. Flight Controller Block Diagram

Adapted from [19], the command orientation is the vector

$$\Phi_{out} = \begin{bmatrix} \theta_{out} \\ \phi_{out} \\ \psi_{out} \end{bmatrix} \tag{41}$$

with components

$$\theta_{out} = \tan^{-1}\left(\frac{F_x \cos(\psi) + F_y \sin(\psi)}{F_z} \right), \tag{42}$$

$$\phi_{out} = \tan^{-1}\left(\frac{F_x \sin(\psi) - F_y \cos(\psi)\cos(\theta_{out})}{F_z} \right), \tag{43}$$

and ψ_{out} set to a constant zero. This eliminates quadrotor yaw through the course of flight. Command thrust is given by

$$f_t = -\frac{F_z}{\cos(\theta_{out})\cos(\phi_{out})}. \tag{44}$$

D. 3D VISUALIZER

The 3D visualizer animates the intercept in a virtual environment. The visualizer uses a computer aided design (CAD) model and Simulink VR toolbox. The target

position, quadrotor position, and quadrotor orientation are inputs to the Simulink VR sink as shown in Figure 11.

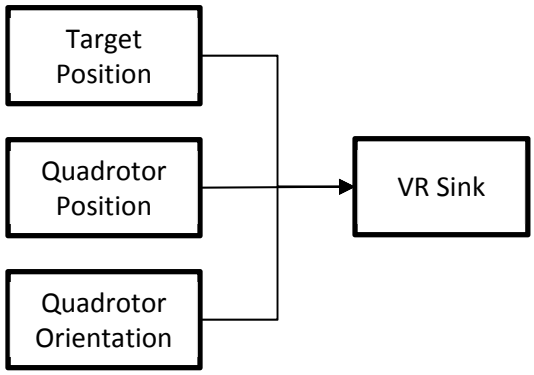


Figure 11. 3D Visualizer Block Diagram

The quadrotor and target represented in a virtual world are shown in Figure 12. The virtual world is a 200 m square, so the size of the quadrotor and target are exaggerated. The target is represented by a red sphere, and the quadrotor is represented by a CAD model.

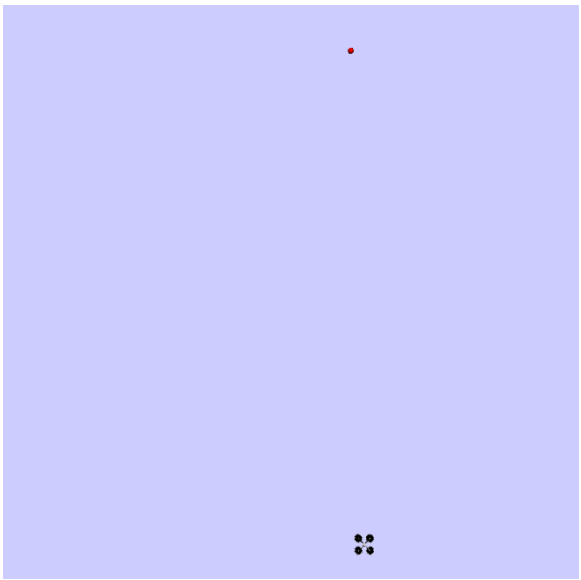


Figure 12. 3D Visualizer Virtual World

The quadrotor CAD model used by the visualizer is shown in Figure 13. The model is based on a typical quadrotor configuration.

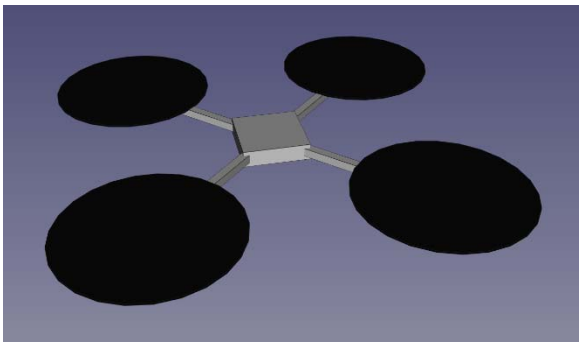


Figure 13. CAD Model of Quadrotor

VI. SIMULATION DESIGN

A. OVERVIEW

The purpose of the simulation is to compare the performance of each of the trajectory planners. The simulation produces an intercept trajectory which is flown by a simulated quadrotor. The energy used by the simulated quadrotor and intercept time are recorded. Motor power consumption rates for a hypothetical quadrotor, measured and listed in Appendix A, are applied to the simulation to estimate the total power consumption per flight. Based on [4] and [5], some planners may have different performance based on the intercept geometry; therefore, each planner is evaluated in three types of intercept geometries. This is achieved by varying target trajectory start location, heading, and velocity.

B. SIMULATION ASSUMPTIONS

Some assumptions are made to narrow the scope of the simulation. The simulation assumes that the trajectory planner has perfect knowledge of the target and interceptor position. It also assumes that the target has constant velocity and heading. Additionally, an intercept is considered to occur when the distance between the trajectory and target is below the intercept threshold of five meters. This threshold is arbitrary for this experiment but should be adjusted based on required distance to deploy payload at intercept. After an intercept occurs, the simulation is complete, and target tracking after intercept is not evaluated.

C. DEFINITION OF FLIGHT PERFORMANCE MEASURE

Two metrics are used to compare the performance of each trajectory. First, the total energy spent to achieve intercept is calculated. This is the energy used by the simulated quadrotor as it flies the command trajectory. Second, the time to achieve intercept is recorded. This is the time that the distance between the target and command trajectory is less than the intercept threshold. The results are compared and evaluated to determine which trajectory planner is fastest and which uses the least energy.

D. INTERCEPT GEOMETRY TYPES

The simulated environment is a three-dimensional 200-m arena. This allows the simulation to span a realistic distance for an air-to-air intercept. The x and y axes are of most interest, and for clarity the results are displayed in the x-y plane.

Three types of intercept geometry are simulated with a non-maneuvering target. This means that the target maintains constant velocity without change in heading. Investigating the performance of the trajectory planners against a maneuvering target is left for future work.

Crossing, head-on, and tail-chase geometry for a non-maneuvering target are shown in Figure 14. The target is represented in red, and the interceptor is represented in blue.

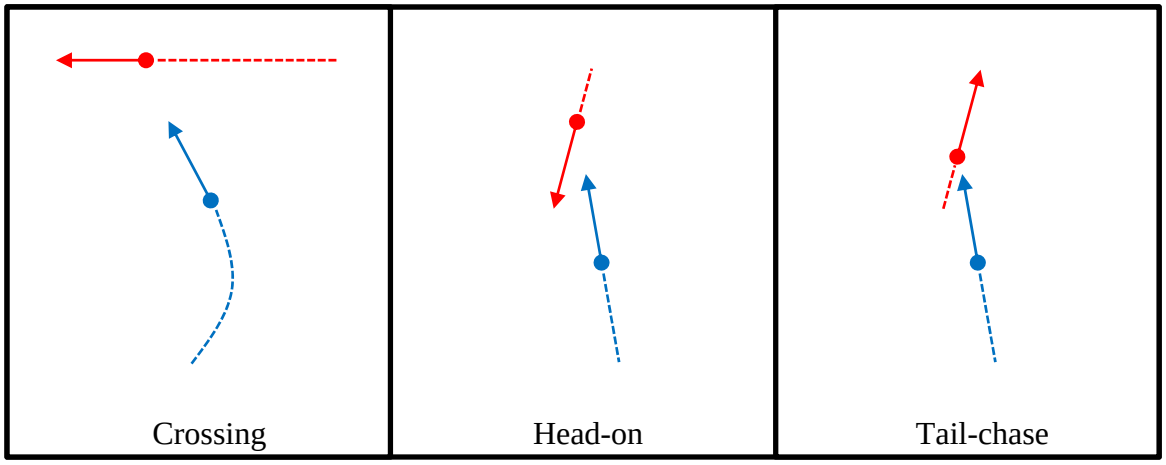


Figure 14. Geometry Types

The starting positions and headings for the target and interceptor for each geometry are listed in Table 2. All headings are measured from North (x-axis) and positions are in meters with the format (x, y, z). In all cases, the target has a constant velocity of seven meters per second, and the interceptor trajectory starts at rest at the same height above ground as the target. The simulated quadrotor starts at the ground

$(z = 0)$ but at the same x-y position as the interceptor trajectory, thereby including takeoff in the simulated flight.

Table 2. Target and Interceptor Starting Positions and Headings

	Crossing	**Head-on**	**Tail-chase**
Target	(50, 100, 2), 270°	(100, 25, 2), 195°	(-50, -25, 2), 020°
Interceptor	(-100, 25, 0)	(-100, 25, 0)	(-100, 25, 0)

E. SIMULATED EXPERIMENTAL SETUP

To compare the performance of the trajectory planners in simulation, the experiment was implemented as a MATLAB function. The inputs to the experiment function were interceptor starting position, target starting position, target speed, and target heading. When the experiment function is called, it runs the Simulink experimental environment described in Figure 7. When the Simulink simulation is complete, the experiment function saves the trajectory plot and flight data produced by the simulation. The function returns the intercept time and energy used. A MATLAB script calls the experiment function for each trajectory planner and geometry type, totaling nine experiments. The script then plots the time and energy used for each experiment for comparison.

The results of the nine experiments are then compared to the baseline (LQT) intercept results described in the Optimal Trajectories section. The LQT results are computed using a single MATLAB script that solves Equations (32) through (37) to compute an optimal intercept trajectory. The trajectory is saved and loaded into a look-up table in the experimental environment acting in place of the trajectory planner block. An experiment is run for each geometry type using the LQT trajectories, which allows the on-line trajectories to be compared to the optimal LQT trajectory.

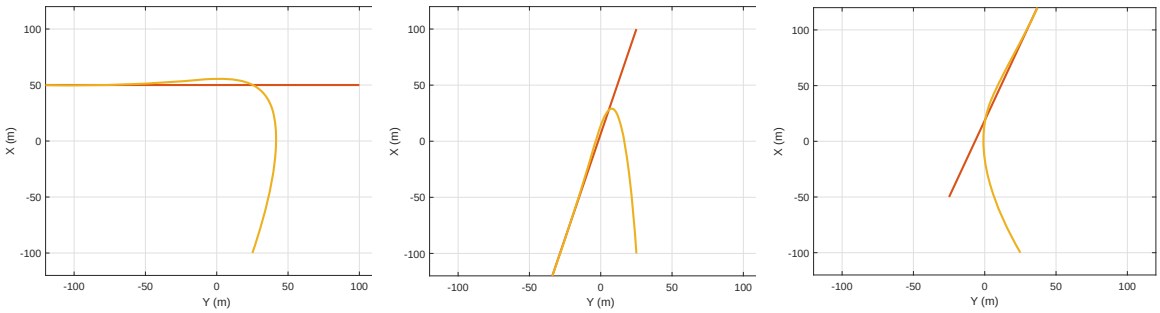
THIS PAGE INTENTIONALLY LEFT BLANK

VII. SIMULATION RESULTS

Simulation results for the three on-line trajectory planners are presented individually, organized by intercept geometry first, then by planner type. The LQT results, simulated by a quadrotor as if they were generated in real time, are presented following the on-line results. All results are then compiled into a single figure for comparison.

The results for each experiment include a trajectory plot and simulated quadrotor flight statistics. Each trajectory plot shows the target trajectory (red), the command trajectory (yellow), and the trajectory flown by the simulated quadrotor (blue). The flight statistics show velocity of the quadrotor (blue), interceptor (red), and the command trajectory position error. The closest point of approach (CPA) is marked on the command trajectory position error by a black star.

The optimal intercept (LQT solution) for each geometry is shown in Figure 15. The grid spacing represents 50 m, and the start positions are listed in Table 2. For example, the crossing geometry shows the target starting at the position $(50,100,2)$ and flying west. The interceptor trajectory is shown starting at the position $(-100,25,0)$ and flying north towards the target, gradually turning and following to the west. These off-line results are presented for comparison to the on-line results.



Crossing Head-on Tail-chase

Figure 15. Optimal LQT Trajectories for Each Geometry

The trajectory plot for the MPC planner with crossing geometry is shown in Figure 16. The command trajectory does not immediately turn towards the target. This turn delay corresponds to the number of forward-looking time steps scheduled for the MPC planner. Although the first leg of the intercept is problematic, the flight path following the turn is acceptable. The quadrotor tracks the command trajectory with ease, and the geometry eventually resembles a tail-chase.

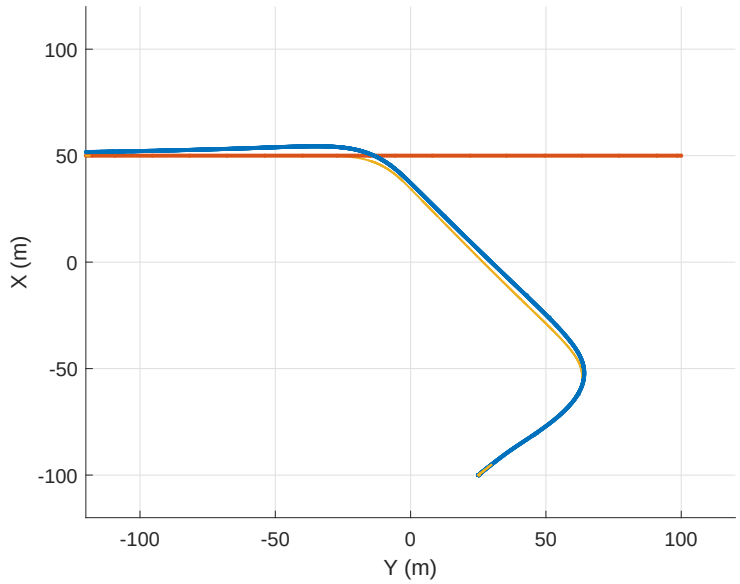


Figure 16. Trajectory Plot of MPC Planner and Crossing Geometry

The flight statistics for the MPC planner in a crossing geometry are shown in Figure 17. The CPA occurs slightly after 20 s, then the position error remains constant at 15 m for the remainder of the simulation. Because the position error does not drop below the intercept threshold of five meters, this simulation does not meet the intercept criteria.

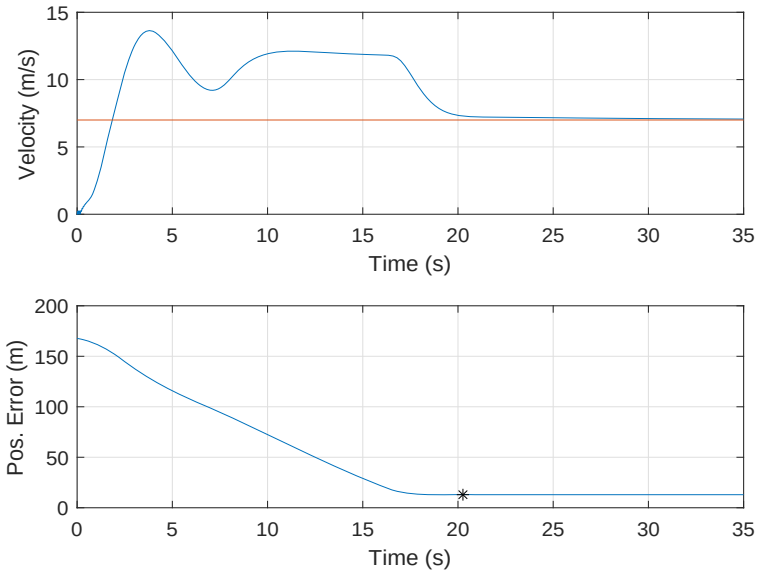


Figure 17. Flight Statistics for MPC Planner and Crossing Geometry

The trajectory plot for the PN planner with crossing geometry is shown in Figure 18. The command trajectory produces an acceptable intercept.

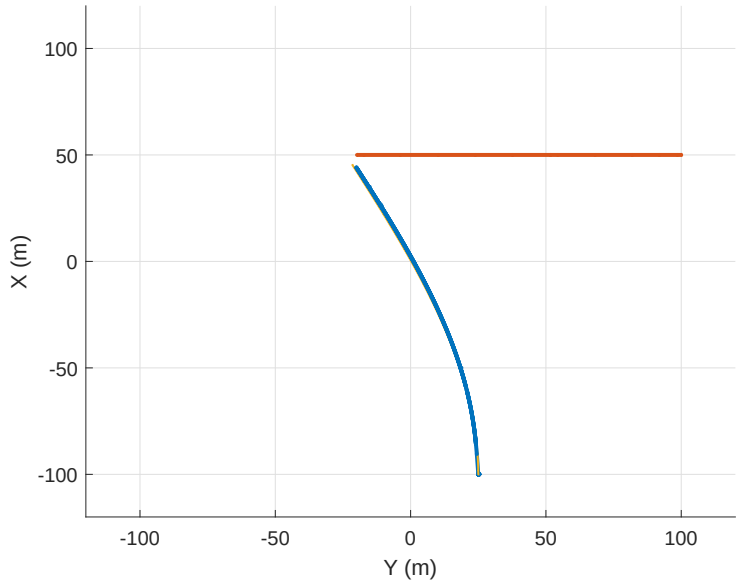


Figure 18. Trajectory Plot of PN Planner and Crossing Geometry

The flight statistics for the PN planner in a crossing geometry are shown in Figure 19. The CPA occurs at the time of intercept, approximately 17 s.

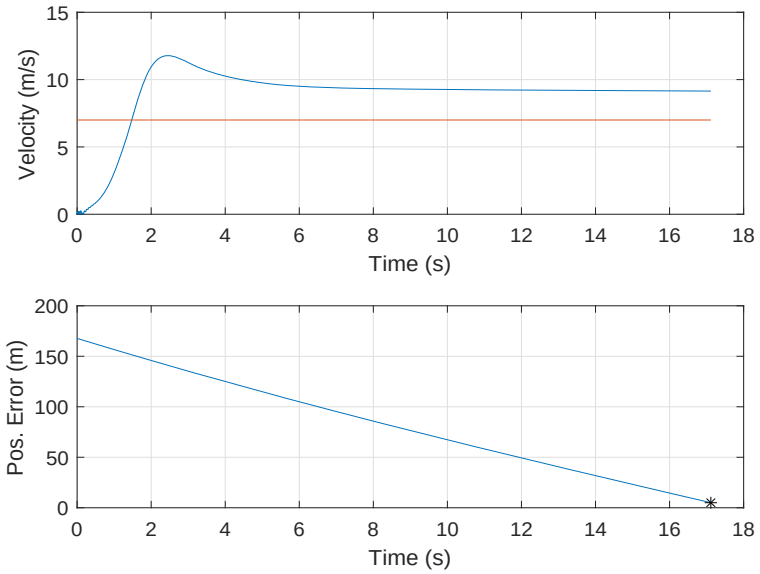


Figure 19. Flight Statistics of PN Planner and Crossing Geometry

The trajectory plot for the PG planner with crossing geometry is shown in Figure 20. The command trajectory produces an acceptable intercept, and the simulated quadrotor exhibits slight tracking error through the turn.

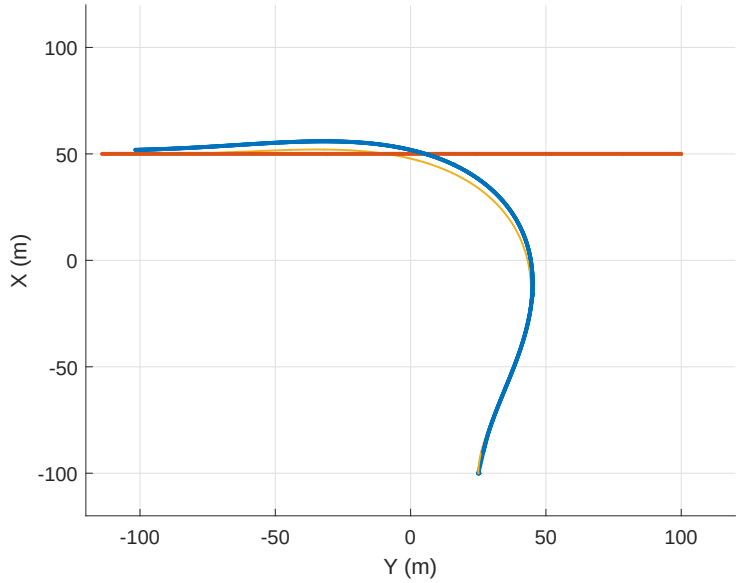


Figure 20. Trajectory Plot of PG Planner and Crossing Geometry

The flight statistics for the PG planner in a crossing geometry are shown in Figure 21. The CPA occurs at the time of intercept, approximately 31 s. In this scenario, the geometry begins as a crossing situation. As the interceptor maneuvers, the geometry begins to look like a tail-chase, and the closing rate decreases. This geometry shift is observed as a change in slope in the position error plot, occurring at approximately 13 s.

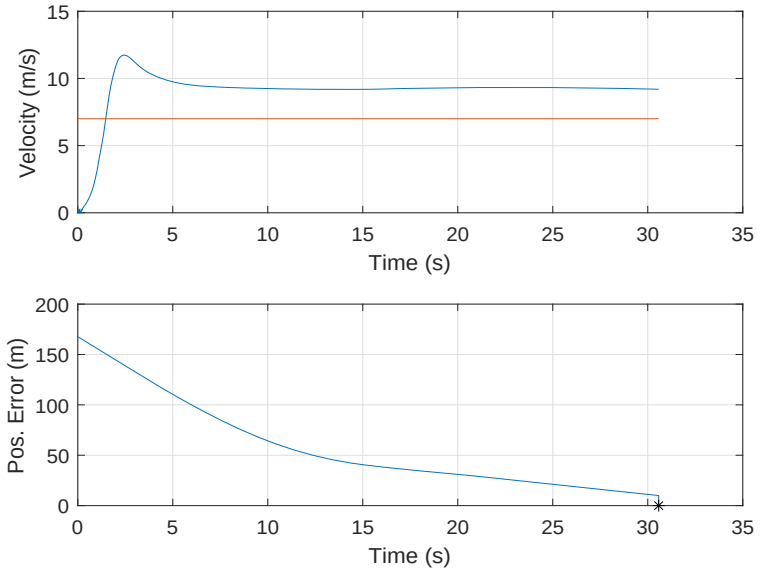


Figure 21. Flight Statistics of PG Planner and Crossing Geometry

The trajectory plot for the MPC planner with head-on geometry is shown in Figure 22. The command trajectory produces an acceptable intercept despite a slight perturbation during the first half of the flight. This disturbance is also observed in Figure 16. In both cases, the disturbance is overcome after the controller has more time observing the system. Other than the disturbance, the flight path is generally direct and the intercept is not affected by steady-state error.

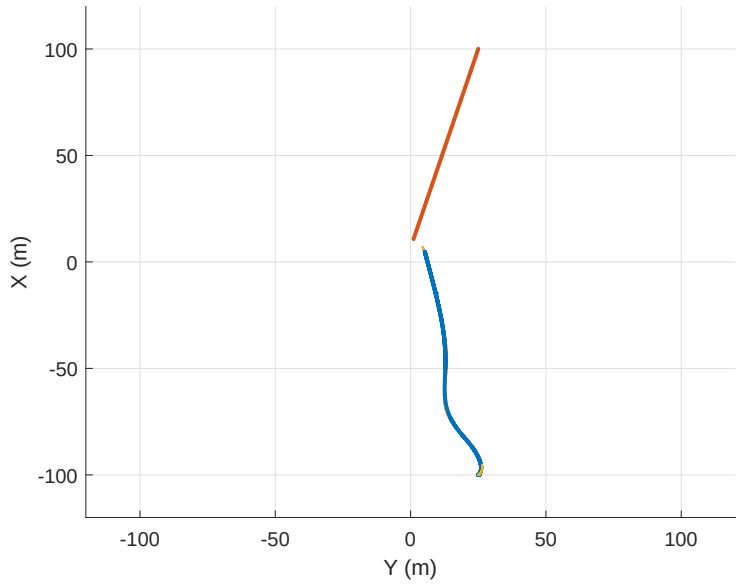


Figure 22. Trajectory Plot of MPC Planner and Head-on Geometry

The flight statistics for the MPC planner in a head-on geometry are shown in Figure 23. The CPA occurs at the time of intercept, at approximately 13 s.

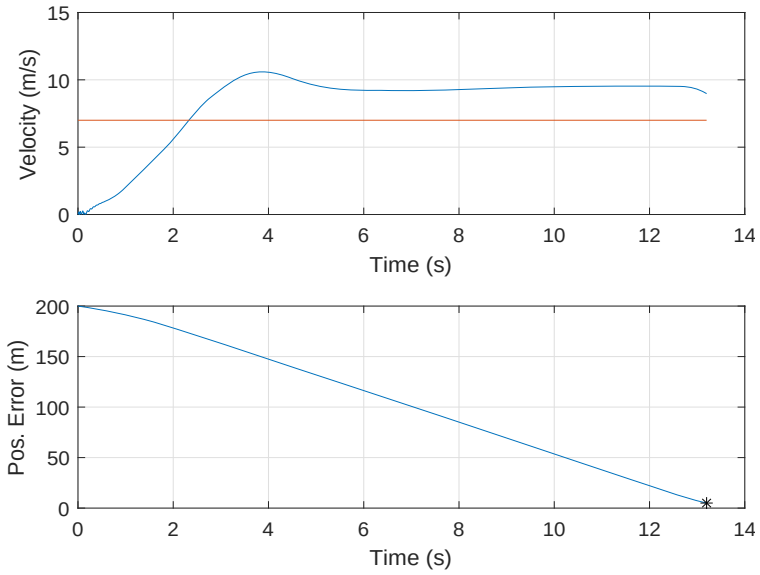


Figure 23. Flight Statistics of MPC Planner and Head-on Geometry

The trajectory plot for the PN planner with head-on geometry is shown in Figure 24. The command trajectory produces an acceptable intercept, and the simulated quadrotor shows minimal tracking error.

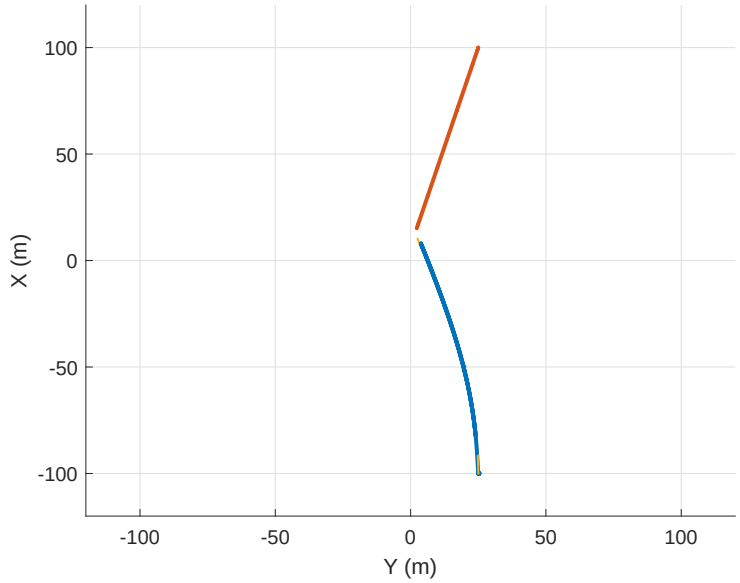


Figure 24. Trajectory Plot of PN Planner and Head-on Geometry

The flight statistics for the PN planner in a head-on geometry are shown in Figure 25. The CPA occurs at the time of intercept, at approximately 13 s. The slope of the position error is constant, indicating that very little maneuvering occurs throughout the flight. This also reflects the direct nature of the flight path, highlighting a potential strength of the PN method.

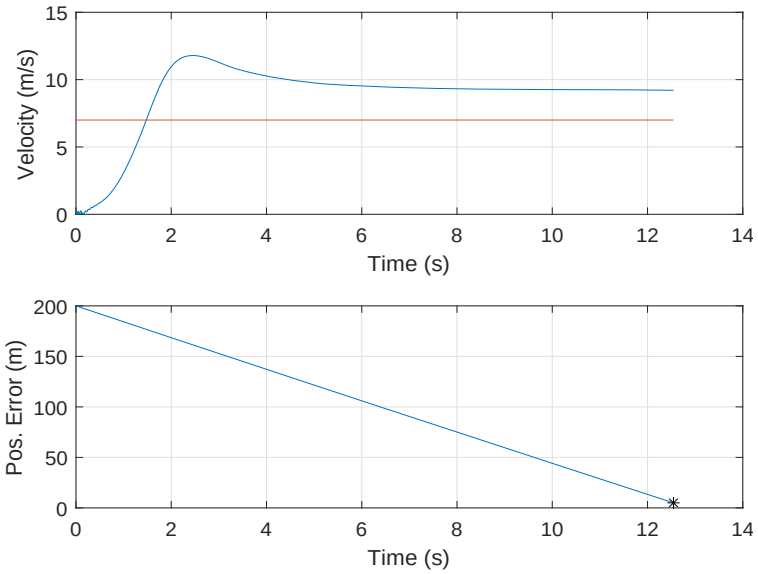


Figure 25. Flight Statistics of PN Planner and Head-on Geometry

The trajectory plot for the PG planner with head-on geometry is shown in Figure 26. The command trajectory produces an acceptable intercept, and the simulated quadrotor shows minimal tracking error.

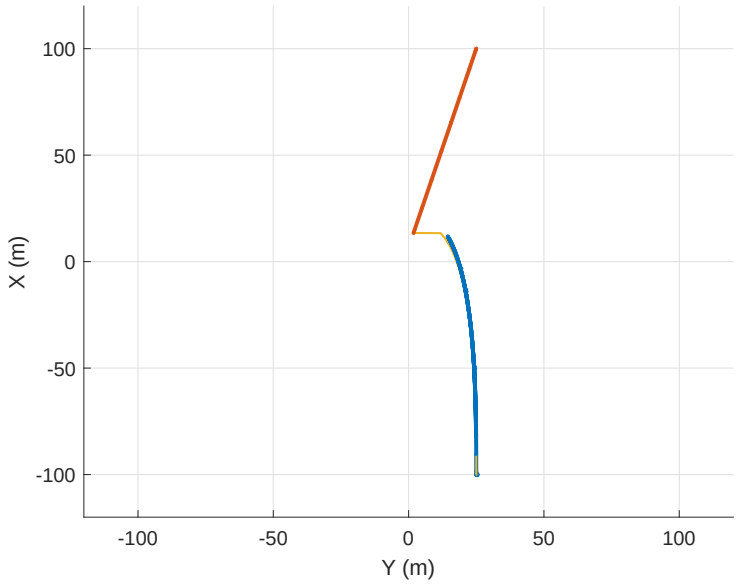


Figure 26. Trajectory Plot of PG Planner and Head-on Geometry

The flight statistics for the PG planner in a head-on geometry are shown in Figure 27. The CPA occurs at the time of intercept, at approximately 13 s. The slope of the position error is constant until the last second of intercept. This change corresponds to a turn in the flight path. Although this qualifies as an intercept, the position of the trajectory lies to the side of the target instead of ahead. This may be less ideal depending on the payload.

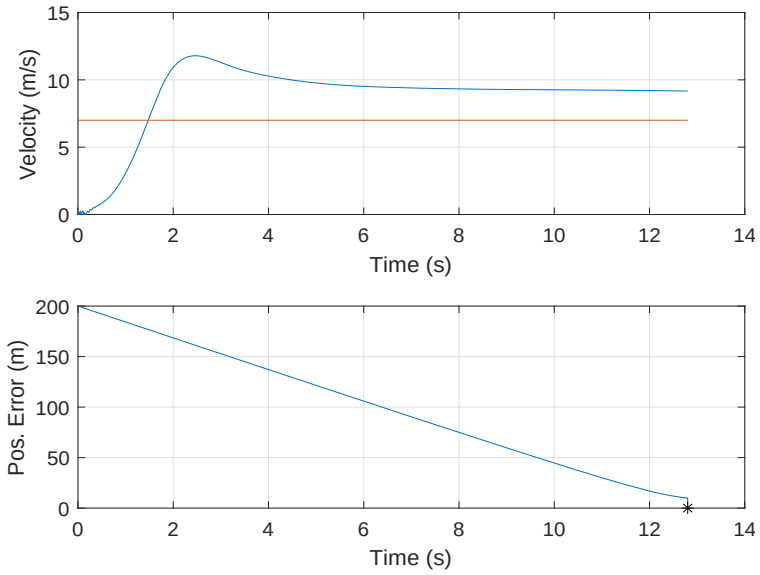


Figure 27. Trajectory Plot of PG Planner and Head-on Geometry

The trajectory plot for the MPC planner with tail-chase geometry is shown in Figure 28. The simulated quadrotor shows some tracking error through the turn. After the turn, the scenario becomes a tail-chase. As with the crossing geometry simulation, a steady-state tracking error occurs, and the command trajectory fails to reach the target.

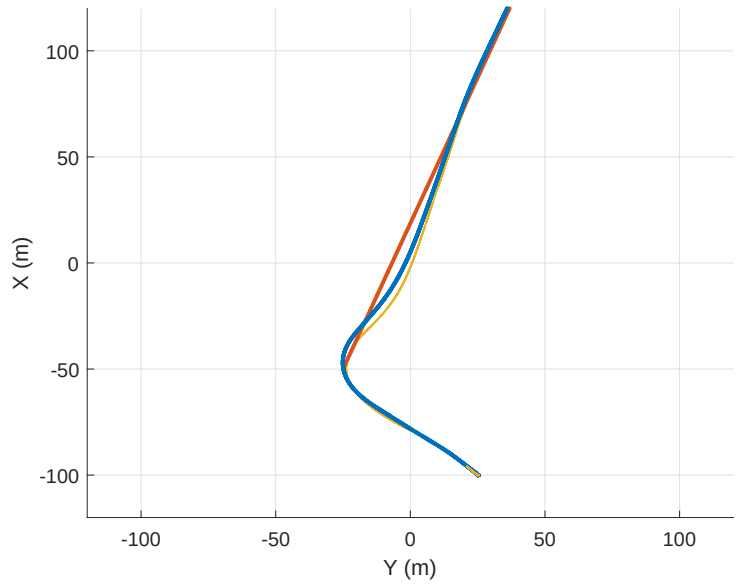


Figure 28. Trajectory Plot of MPC Planner and Tail-chase Geometry

The flight statistics for the MPC planner in a tail-chase geometry are shown in Figure 29. The CPA occurs just before 25 s, and the position error becomes a constant of approximately 17 m. This does not produce an acceptable intercept because the steady-state error is above the intercept threshold.

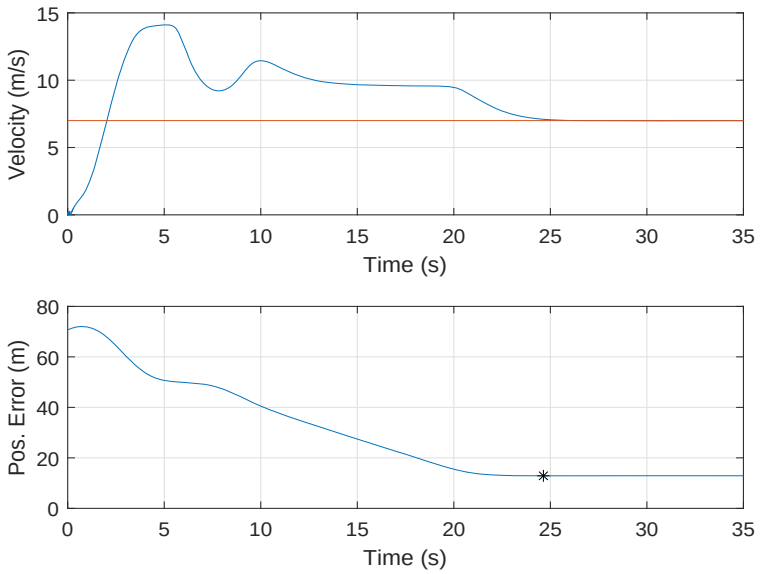


Figure 29. Flight Statistics of MPC Planner and Tail-chase Geometry

The trajectory plot for the PN planner with tail-chase geometry is shown in Figure 30. This straight-line flight path shows no maneuvering, thereby producing a direct intercept. These results highlight the effectiveness of the PN trajectory planner because there is no significant turning required by the simulated quadrotor. This produces minimal tracking error and minimal energy wasted in turning, making the maneuver efficient.

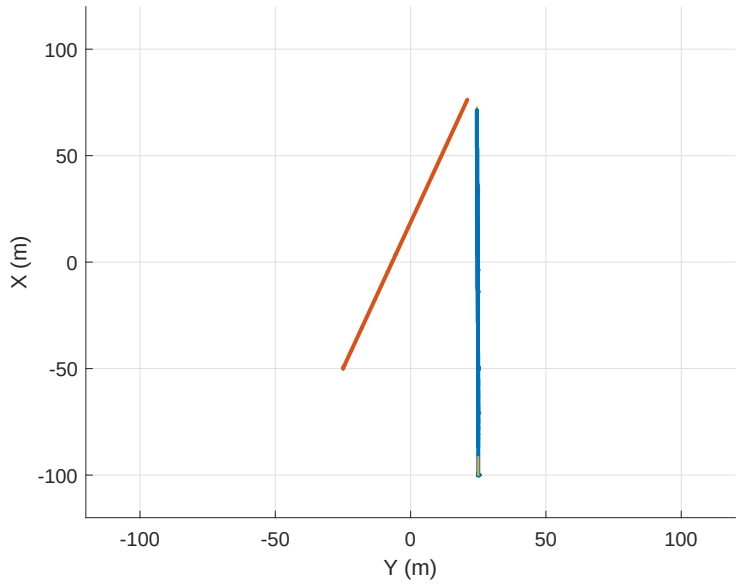


Figure 30. Trajectory Plot of PN Planner and Tail-chase Geometry

The flight statistics for the PN planner in a tail-chase geometry are shown in Figure 31. The CPA occurs at the time of intercept, approximately 19 s. The direct flight path is reflected by a constant slope in the position error plot. These results could only be made more efficient by increasing the speed of the command trajectory.

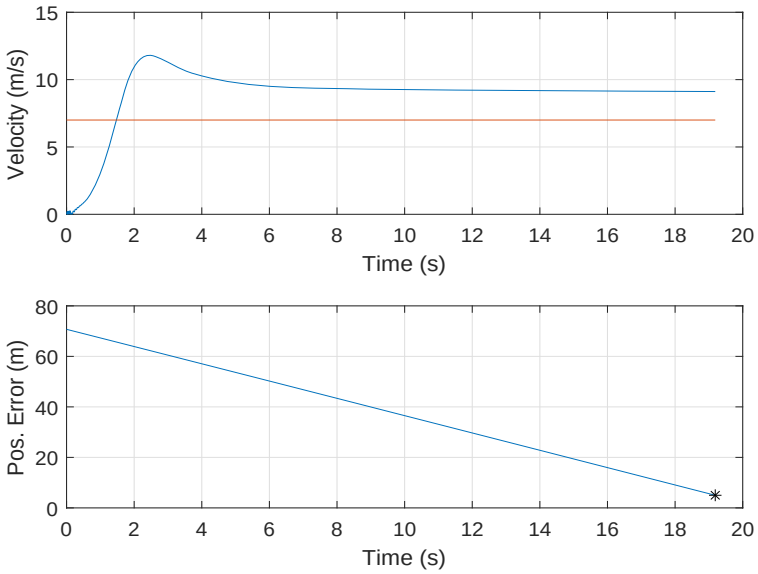


Figure 31. Flight Statistics of PN Planner and Tail-chase Geometry

The trajectory plot for the PG planner with tail-chase geometry is shown in Figure 32. The simulated quadrotor shows some tracking error through the turn.

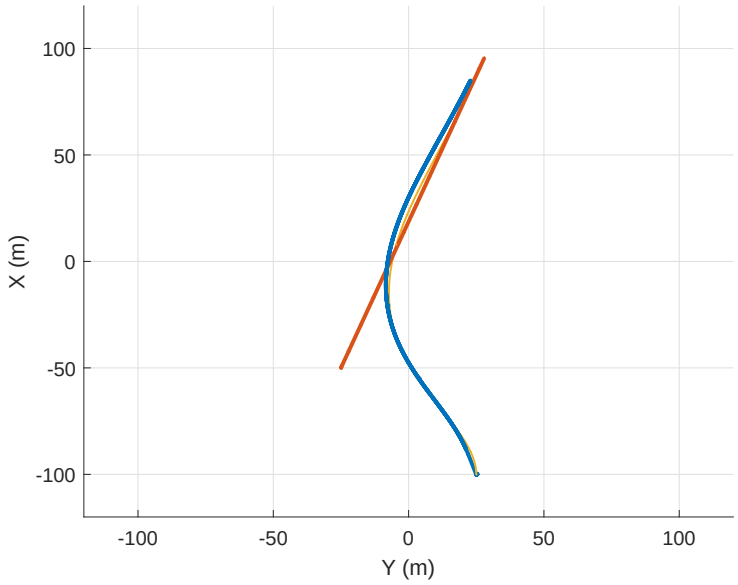


Figure 32. Trajectory Plot of PG Planner and Tail-chase Geometry

The flight statistics for the PG planner in a tail-chase geometry are shown in Figure 33. The CPA occurs at the time of intercept, approximately 22 s.

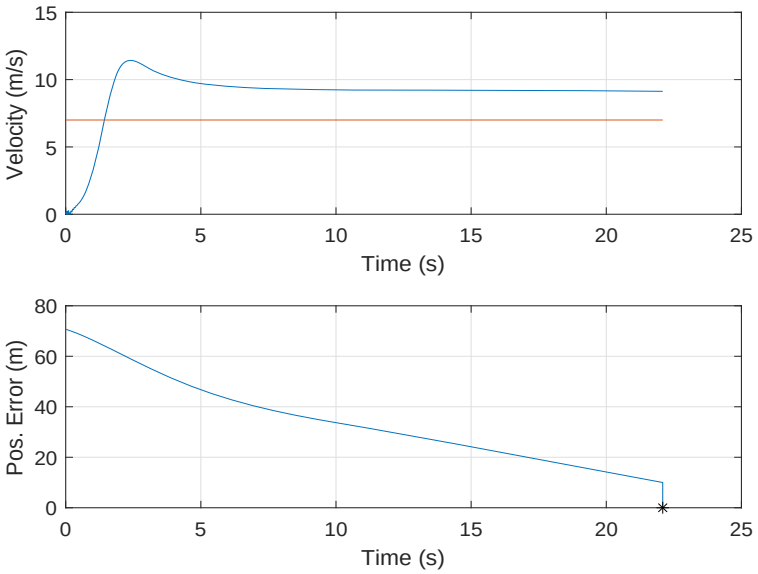


Figure 33. Flight Statistics of PG Planner and Tail-chase Geometry

The trajectory plot for the LQT solution for crossing geometry is shown in Figure 34. The simulated quadrotor shows some tracking error through the turn.

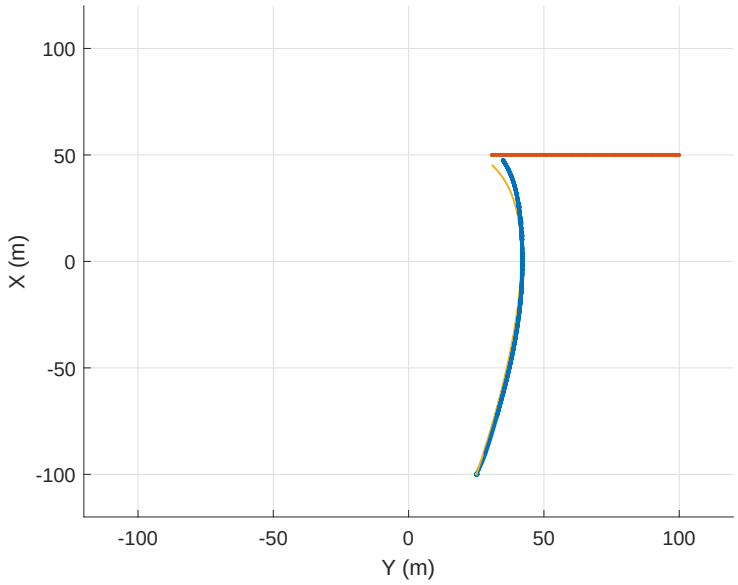


Figure 34. Trajectory Plot of LQT Solution and Crossing Geometry

The flight statistics for the LQT solution for crossing geometry are shown in Figure 35. Quadrotor velocity reaches the simulation limits established by the flight controller. The CPA occurs at the time of intercept, approximately 10 s. The slope of the position error changes throughout the intercept, reflecting the course and speed changes made in the flight path.

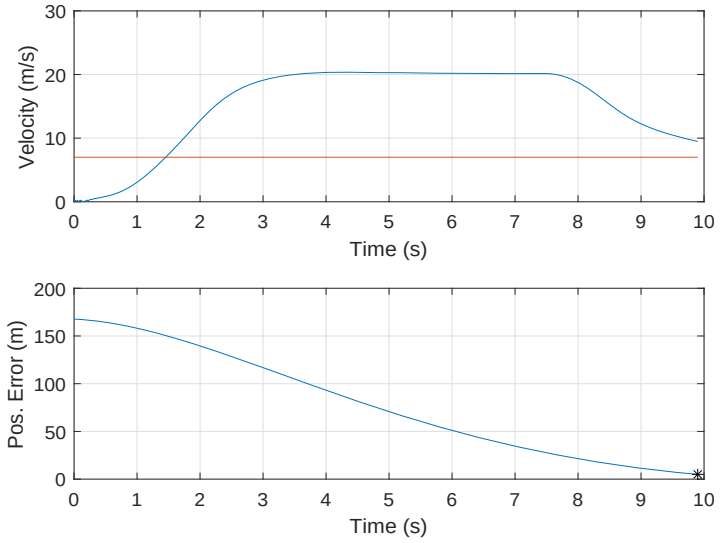


Figure 35. Flight Statistics of LQT Solution and Crossing Geometry

The trajectory plot for the LQT solution for head-on geometry is shown in Figure 36. The simulated quadrotor shows some tracking error through the turn immediately before intercept. The flight path for this optimal solution is not as direct as the PN results, but the end position of the trajectory is in the preferred location ahead of the target.

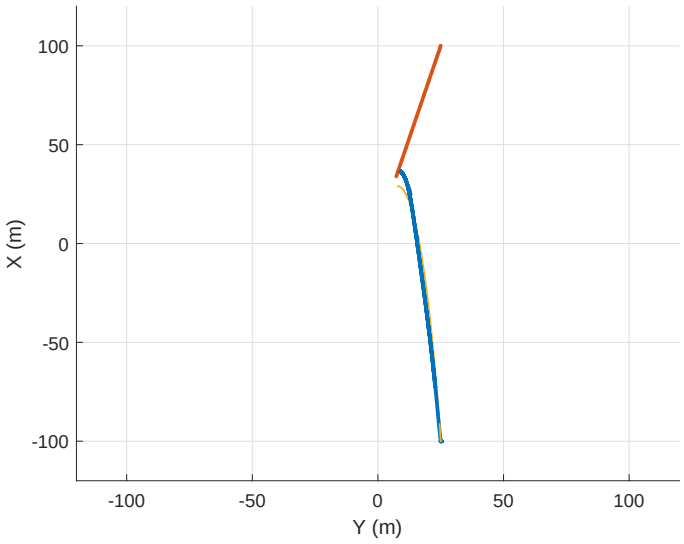


Figure 36. Trajectory Plot of LQT Solution and Head-on Geometry.

The flight statistics for the LQT solution for head-on geometry are shown in Figure 37. The CPA occurs at the time of intercept, approximately 10 s.

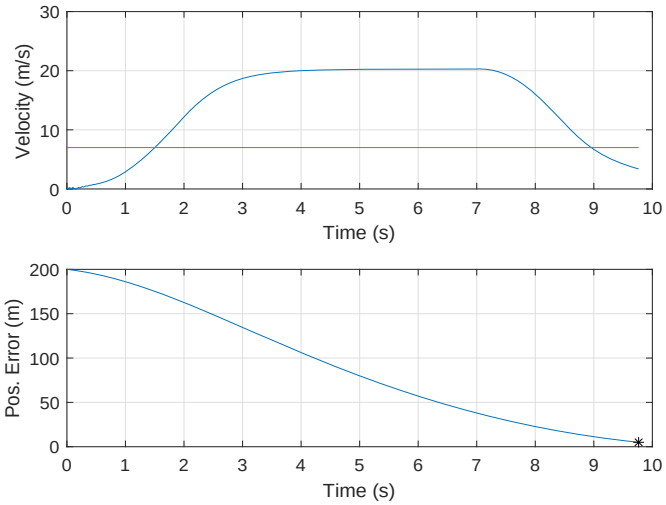


Figure 37. Flight Statistics of LQT Solution and Head-on Geometry

The trajectory plot for the LQT solution for tail-chase geometry is shown in Figure 38. The simulated quadrotor shows some tracking error through the turn immediately before intercept. The flight path shows some maneuvering but is generally a direct intercept.

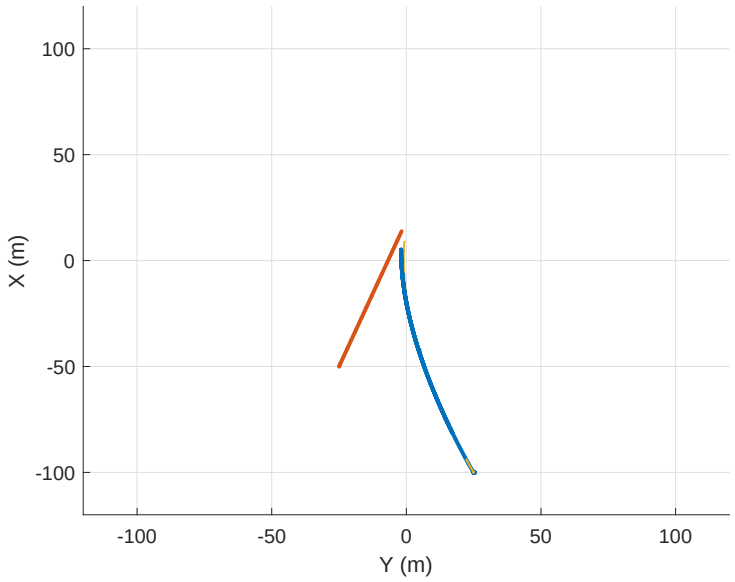


Figure 38. Trajectory Plot of LQT Solution and Tail-chase Geometry

The flight statistics for the LQT solution for tail-chase geometry are shown in Figure 39. The CPA occurs at the time of intercept, approximately 10 s.

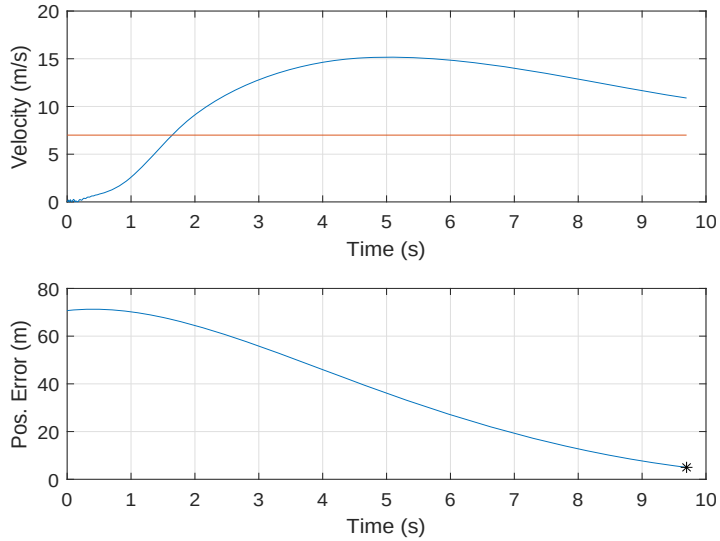


Figure 39. Flight Statistics of LQT Solution and Tail-chase Geometry

The trajectory plots are combined and displayed in Figure 40. The axes and scale are omitted but unchanged from their representation in Figure 16 through Figure 38.

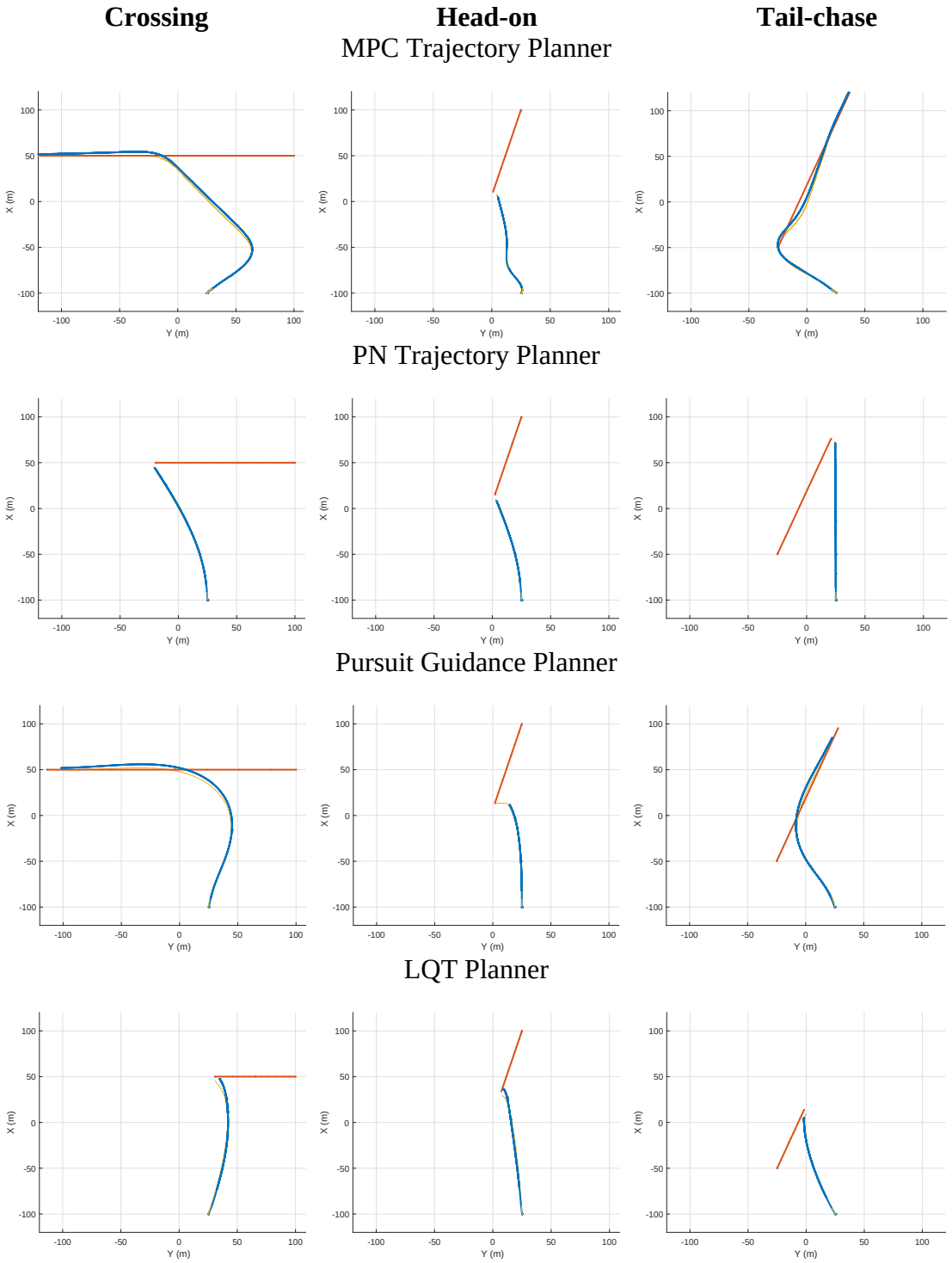


Figure 40. Combined Results

The time to intercept (seconds) for each planner and geometry type are shown in Table 3.

Table 3. Experimental Intercept Times

	MPC	PN	PG	LQT
Crossing	20.3	17.1	30.6	9.9
Head-on	13.2	12.5	12.8	9.8
Tail-chase	24.6	19.2	22.1	9.7

The energy used (joules) by the simulated quadrotor during the experimental intercept for each planner and geometry type are shown in Table 4.

Table 4. Experimental Intercept Energy

	MPC	PN	PG	LQT
Crossing	2931.5	1440.9	2546.1	930.6
Head-on	1118.5	1068.8	1091.0	941.5
Tail-chase	2928.8	1608.8	1848.8	837.7

A summary of experimental intercept times and total energy is shown in Figure 41. Results are grouped by geometry type and colored based on trajectory planner.

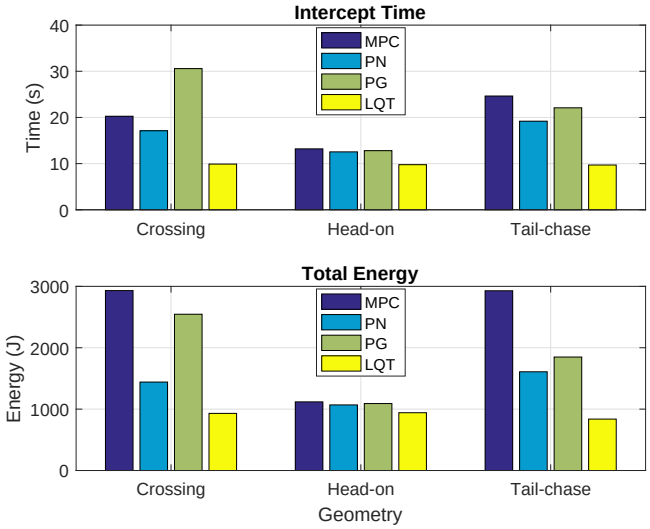


Figure 41. Summary of Experimental Data

VIII. CONCLUSION AND FUTURE WORK

A. CONCLUSION

The research objective was to determine if optimal control can outperform classic missile control methods when applied to quadrotor drones. This is achieved by simulating the intercept of a target with a quadrotor drone and comparing the performance measure of each trajectory planner. The trajectory planner with the best performance measure in simulation represents the best planner for actual flight trajectory planning.

A set of optimal but off-line intercept trajectories are calculated as a benchmark. Each of the on-line trajectory planners (PG, PN, and MPC) executes an intercept maneuver while the time and energy spent by a simulated quadrotor are recorded. Then, the benchmark optimal trajectories are flown by the simulated quadrotor, and the same time and energy data are recorded. A comparison of the trajectories and a summary of the flight statistics highlight the strengths and weaknesses of each planner.

The optimal results when flown by the simulated quadrotor indeed show the highest intercept performance. The intercept occurs in all three geometries at just before 10 s, with total energy spent to intercept below 1000 J. Because this solution requires knowledge of the flight path of the target over the full simulation time, this method cannot be used to calculate an on-line command trajectory.

The best on-line performance observed is the PN planner. It consistently achieves an intercept in the least amount of time and with the least amount of energy across all geometry types. The low energy consumption is consistent with the minimal maneuvering for each of the intercepts.

The next best performance after PN is both the MPC and PG planners. Their results are mixed based on geometry. For crossing geometry, MPC is faster but uses more energy than PG. This is acceptable in a situation where intercept time is more important than stretching the flight endurance of the aircraft. For head-on geometry, PG is both faster and uses less energy than MPC. The margin, however, is very small—less than a

second and a 100 joule difference. Finally, tail-chase results show that PG uses less time and energy than MPC.

B. FUTURE WORK

The MPC steady state error is the cause of its low performance. Improving the configuration of this planner in future work may produce better results by the MPC trajectory planner.

Target maneuvers are likely to have an impact on the performance of the PN trajectory planner, as suggested by [4], [5], and [7]. The effects of target maneuvers on all guidance types should be investigated in future work.

Additionally, the estimation of energy consumption and intercept time by a simulated quadrotor should be validated with physical implementation. The motor, electronic speed control (ESC), and propeller used in the simulation should be the same as those used on any future real hardware.

APPENDIX A. MOTOR AND PROPELLER THRUST MEASUREMENT

In order to estimate the energy consumption of the simulated quadrotor, the power consumption and force of the motor and propeller combination must be measured at each throttle setting. An apparatus is used to measure the force applied to a sensor by the motor and propeller, while another sensor measures current and voltage used by the motor. The throttle is stepped in five percent increments, holding each for 10.0 s while data is collected. The average of the measurements at each throttle setting is used to produce a data point.

The force and power measurement apparatus is shown in Figure 42. The motor and propeller are attached to a lever arm balanced with a counterweight. When the motor produces a force, it is measured by the force sensor. The motor ESC is powered by a digital power supply which records the voltage, current, and power at each throttle setting.

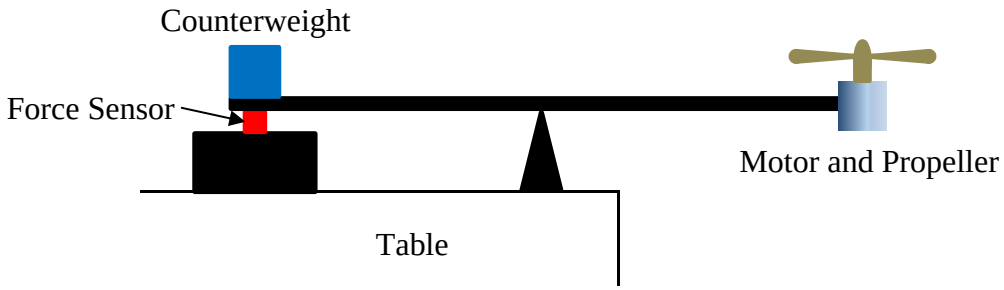


Figure 42. Force and Power Measurement Apparatus

The data collected with the apparatus is displayed in Table 5. The motor tested is a T-Motor MT2208, rated at 1100 kv, with a 10×4.5 APC propeller.

Table 5. Motor and Propeller Performance Measurements

Throttle	Voltage (V)	Current (A)	Power (W)	Thrust (g)
0%	11.99	0.10	1.239	0
5%	11.98	0.10	1.241	4
10%	11.98	0.18	2.136	17
15%	11.98	0.29	3.457	36
20%	11.98	0.43	5.101	57
25%	11.98	0.59	7.03	77
30%	11.98	0.76	9.096	99
35%	11.97	0.95	11.361	123
40%	11.97	1.16	13.935	146
45%	11.97	1.38	16.476	167
50%	11.97	1.57	18.807	187
55%	11.96	1.83	21.868	211
60%	11.96	2.22	26.486	244
65%	11.95	2.68	31.981	285
70%	11.95	3.21	38.334	329
75%	11.94	3.80	45.332	372
80%	11.93	4.46	53.206	419
85%	11.92	5.18	61.771	466
90%	11.92	6.05	72.133	518
95%	11.90	7.03	83.616	574
100%	11.90	7.47	88.876	599

APPENDIX B. SIMULINK AND MATLAB CODE

The supporting code for the Simulink model and the LQT MATLAB script are presented in this appendix.

A. EXPERIMENTAL FUNCTION, EMBEDDED MATLAB FUNCTIONS, AND DATA PLOTS

The scripts and functions in this section are used in the Simulink model.

1. Main Script

```
% Simulate a quadrotor intercepting a moving target using different methods
% of trajectory generation. In real time, generate intercept trajectories
% and simulate a quadrotor flying to the command trajectory. This script
% runs the experimental simulation for each geometry type and trajectory
% generator.

% At the end of the simulation calculate and display:
% 1. Plot of target and command trajectory and path flown by simulated acft
% 2. Flight statistics (velocity, distance to tgt)
% 3. 3d view of aircraft availaible in simulink model

% By Rob Allen
% --------------------------------------------------------------------------
clc
clear all
close all

mpcinit;
open_system('Quad_intercept')

% Select trajectory planner for intercept
% 1 = MPC finite horizion
% 2 = LQR infinite horizion
% 3 = Proportional Navigation
% 4 = Pursuit guidance

% initialize geometry
% 1 = crossing
% 2 = head on
% 3 = tail chase

% Run experiment for each geometry and each trajectory planner
for geometry = 1:3
  for trj_sel = 1:4
```

```
switch geometry
  case 1 %crossing
    % set target initial conditions
    V = 7;
    heading = -90;
    Vx = V*cosd(heading);
    Vy = V*sind(heading);
    t_x0 = [50 100 -2 Vx Vy 0];

    % set Interceptor initial conditions
    i_x0 = [-100 25 0];

    % set Interceptor trajectory initial conditions
    it_x0 = [-100 25 -2];

    %load optimal trajectory (only used by trajectory 4)
    load('optimal_crossing.mat')
  case 2 % head on
    % set target initial conditions
    V = 7;
    heading = -165;
    Vx = V*cosd(heading);
    Vy = V*sind(heading);
    t_x0 = [100 25 -2 Vx Vy 0];

    % set Interceptor initial conditions
    i_x0 = [-100 25 0];

    % set Interceptor trajectory initial conditions
    it_x0 = [-100 25 -2];

    %load optimal trajectory (only used by trajectory 4)
    load('optimal_headon.mat')
  case 3 % tail chase
    % set target initial conditions
    V = 7;
    heading = 20;
    Vx = V*cosd(heading);
    Vy = V*sind(heading);
    t_x0 = [-50 -25 -2 Vx Vy 0];

    % set Interceptor initial conditions
    i_x0 = [-100 25 0];

    % set Interceptor trajectory initial conditions
    it_x0 = [-100 25 -2];

    %load optimal trajectory (only used by trajectory 4)
    load('optimal_tailchase.mat')
  otherwise
end
```

```
    [ tcpa(geometry, trj_sel), cpa(geometry, trj_sel), tenergy(geometry, trj_sel) ] =
experiment(trj_sel, geometry);
  end
end

% Plot final experiment statistics

plot_experimental_results;

filename = 'output\testdata.xlsx';
xlswrite(filename,tenergy,1,'B2:E4')
xlswrite(filename,tcpa,1,'B8:E10')
save('output\results');
```

2. Experiment Function

```
function [ tcpa, cpa, tenergy ] = experiment( planner, geometry)
% experiment(planner, geometry) runs one quadrotor simulation with the given
% planner and geometry type. Execute the Quad_intercept.slx simulation and
% return time to CPA, CPA and Total Energy.
% Trajectory Planners are defined in the simulink model by:
% 1 = MPC finite horizion
% 2 = LQR infinite horizion
% 3 = Proportional Navigation
% 4 = Pursuit guidance

% Simulate a quadrotor intercepting a moving target using different methods
% of trajectory generation. In real time, generate intercept trajectories
% and simulate a quadrotor flying to the command trajectory.

% At the end of the simulation calculate and display:
% 1. Plot of target and command trajectory and path flown by simulated acft
% 2. Flight statistics (velocity, distance to tgt, power and energy
% 3. 3d view of aircraft availaible in simulink model

% By Rob Allen
% --------------------------------------------------------------------

trj_sel = planner;

sim('Quad_intercept')

axislimits = 120;

v = [state(:,4) state(:,5) state(:,6)];
pos = [state(:,1) state(:,2) state(:,3)];

% To display simulation in the NED frame, format plot3(y, x, -z)
figure(1)
plot3(-pos(:,2),-pos(:,1),-pos(:,3), 'LineWidth', 2)
```

```
hold on
plot3(-pos_t(:,2),-pos_t(:,1),-pos_t(:,3), 'LineWidth', 2)
plot3(-trj(:,2),-trj(:,1),-trj(:,3), 'LineWidth', 1)
hold off
grid on
axis([-axislimits, axislimits, -axislimits, axislimits, -axislimits, axislimits])
xticklabels({'100','50','0','-50','-100'})
yticklabels({'100','50','0','-50','-100'})
xlabel('Y (m)')
ylabel('X (m)')
zlabel('Z')
view(180, 90)

saveas(gcf,['output/G', num2str(geometry),'P', num2str(planner),'.eps'],'epsc')

v = [state(:,4) state(:,5) state(:,6)];

% Plot and save flight statistics and return flight metrics
% pos_t : position of target
% v : velocity of target
% pos_error : position of target minus position of interceptor
% Ft : Thrust force applied to interceptor
% t : time vector
% n : experiment number
% vtrj : velocity of interceptor trajectory

simname = 'Flight Statistics'; %change to PID guidance later

vel_t = vtrj;
v_i = sqrt(v(:,1).^2 + v(:,2).^2 + v(:,3).^2);
v_t = sqrt(vel_t(:,1).^2 + vel_t(:,2).^2 + vel_t(:,3).^2);

pos_err = sqrt(pos_error(:,1).^2 + pos_error(:,2).^2 + pos_error(:,3).^2);
[CPA, I] = min(pos_err);

% Plot flight statistics
figure(2)
subplot(2,1,1) % Velocity of interceptor and target
plot(t, v_i)
hold on
plot(t(1:length(v_t)), v_t)
hold off
grid on
ylabel('Velocity (m/s)')
xlabel('Time (s)')

subplot(2,1,2) % Position error (target minus trajectory)
plot(t, pos_err)
hold on
plot(t(I), CPA, '*k')
hold off
grid on
```

```
ylabel('Pos. Error (m)')
xlabel('Time (s)')

saveas(gcf,['output/G', num2str(geometry),'P', num2str(planner),'stats.eps'],'epsc')

tcpa = t(I);
cpa = CPA;
tenergy = energy(end);

end
```

3. Initialize MPC Script

```
% Calculate an intercept trajectory of a moving target for a quadrotor
% using simulink MPC block. This file creates the MPC object necessary for
% the simulink block.

% By Rob Allen
% --------------------------------------------------------------------------

%define a system xdot = Ax + Bu, y = Cx + Du
A = [0 0 0 1 0 0;
   0 0 0 0 1 0;
   0 0 0 0 0 1;
   0 0 0 0 0 0;
   0 0 0 0 0 0;
   0 0 0 0 0 0];

B = [0 0 0;
   0 0 0;
   0 0 0;
   1 0 0;
   0 1 0;
   0 0 1];

C = [1 0 0 0 0 0;
   0 1 0 0 0 0;
   0 0 1 0 0 0;
   0 0 0 1 0 0;
   0 0 0 0 1 0;
   0 0 0 0 0 1];

D = [0 0 0;
   0 0 0;
   0 0 0;
   0 0 0;
   0 0 0;
   0 0 0];

states = {'x' 'y' 'z' 'vx' 'vy' 'vz'};
```

```
inputs = {‘fx’ ‘fy’ ‘fz’};
outputs = {‘x’ ‘y’ ‘z’ ‘vx’ ‘vy’ ‘vz’};

%create open loop system
sys_ss = ss(A,B,C,D,’statename’,states,’inputname’,inputs,’outputname’,outputs);

%check controllability
co = ctrb(sys_ss);

% Setup the MPC block and limit input
Ts = 0.1; % Time step
p = 50;  % Prediction Horizion
m = 20;  % Control Horizion
umax = 4; % Max input

mpcobj = mpc(sys_ss, Ts, p, m);
mpcobj.MV = struct(‘Min’,{-umax;-umax;-umax},’Max’,{umax;umax;umax}, ...
  ‘RateMin’,{-1;-1;-1}); %Specify input saturation limits
mpcobj.Weights = struct(‘MV’,[1 1 1],’MVRate’,[1 1 1],’OV’, ...
  [1 1 1 1 1 1]); % define weights on manipulated and controlled variables

% Define limits on velocity by creating a state constraint
vmax = 9;
% mpcobj.OV(4).Min = -vmax;
% mpcobj.OV(4).Max = vmax;
% mpcobj.OV(5).Min = -vmax;
% mpcobj.OV(5).Max = vmax;
% mpcobj.OV(6).Min = -vmax;
% mpcobj.OV(6).Max = vmax;


% Set constraints in the form E*u(k+j)+F*y(k+j)<=G
E = [0 0 0;
   0 0 0;
   0 0 0;
   0 0 0;
   0 0 0;
   0 0 0];
F = [0 0 0 0 0 0;
   0 0 0 0 0 0;
   0 0 0 0 0 0;
   0 0 0 1 0 0;
   0 0 0 0 1 0;
   0 0 0 0 0 1];
G = [0; 0; 0; vmax; vmax; vmax];

% select Equal Concern for Relaxation of constraints
V = [1;1;1;0.001;0.001;0.001];

setconstraint(mpcobj,E,F,G,V);
```

4. Embedded Function “idyn”

```
function [sys, x0, str, ts] = idyn(t,x,u,flag, it_x0)
% Interceptor dynamics for Quad_intercept.slx
% By Rob Allen
% --------------------------------------------------------------------
  if abs(flag) == 1

    w = u(1);

    A = [0 0 0 1 0 0;
       0 0 0 0 1 0;
       0 0 0 0 0 1;
       0 0 0 0 -w 0;
       0 0 0 w 0 0;
       0 0 0 0 0 0];

    sys(1:6) = A*x(1:6);


  elseif flag == 3
    sys(1:6) = x(1:6);

  elseif flag == 4
    sampleTime = 0.1;
    sys = t + sampleTime;

  elseif flag == 0
    sizes = simsizes;
    sizes.NumContStates = 6;
    sizes.NumDiscStates = 0;
    sizes.NumOutputs = 6;
    sizes.NumInputs = 1;
    sizes.DirFeedthrough= 1;
    sizes.NumSampleTimes= 1;
    sys = simsizes(sizes);

    % initial conditions
    V = 9;
    heading = 0;
    Vx = V*cosd(heading);
    Vy = V*sind(heading);

    x0 = [it_x0(1) it_x0(2) it_x0(3) Vx Vy 0];
    str = [];
    ts = [ 0 0 ];
  else
    sys = [];
end
```

5. Embedded Function “tdyn”

```
function [sys, x0, str, ts] = tdyn(t,x,u,flag, t_x0)
% Target dynamics for Quad_intercept.slx
% By Rob Allen
% --------------------------------------------------------------------------

  if abs(flag) == 1

    w = u(1);
    w_ = 0;

    R1 = [0 -1 0;
       1 0 0;
       0 0 1];
    R2 = [0 0 -1;
       0 1 0;
       1 0 0];
    R3 = [1 0 0;
       0 1 0;
       0 0 1];

    R = w_*R2+(w*R1);

    A = [0 0 0 1 0 0;
       0 0 0 0 1 0;
       0 0 0 0 0 1;
       0 0 0 R(1,1) R(1,2) R(1,3);
       0 0 0 R(2,1) R(2,2) R(2,3);
       0 0 0 R(3,1) R(3,2) R(3,3)];

    sys(1:6) = A*x(1:6);


  elseif flag == 3
    sys(1:6) = x(1:6);

  elseif flag == 4
    sampleTime = 0.01;
    sys = t + sampleTime;

  elseif flag == 0
    sizes = simsizes;
    sizes.NumContStates = 6;
    sizes.NumDiscStates = 0;
    sizes.NumOutputs = 6;
    sizes.NumInputs = 2;
    sizes.DirFeedthrough= 1;
    sizes.NumSampleTimes= 1;
    sys = simsizes(sizes);
```

```
    % initial conditions
    x0 = t_x0;
    str = [];
    ts = [ 0 0 ];
  else
    sys = [];
end
```

6. Embedded Function “state_eq”

```
function [sys, x0, str, ts] = state_eq(t,x,u,flag, i_x0)
% Quadrotor dynamics for Quad_intercept.slx
% By Rob Allen
% ------------------------------------------------------------------------
  g = 9.8;
  m = 0.812;


  Jx = 0.006;
  Jy = 0.006;
  Jz = 0.012;

  J = [Jx 0 0;
     0 Jy 0;
     0 0 Jz];

  if abs(flag) == 1

    ft = u(1);
    tau = u(2:4);
    phi = x(7);
    theta = x(8);
    psi = x(9);

    w(1:3) = x(10:12);
    w = w';

    Rpsi = [cos(psi) -sin(psi) 0;
        sin(psi)  cos(psi) 0;
        0     0    1];

    Rtheta = [ cos(theta) 0 sin(theta);
          0     1     0 ;
         -sin(theta) 0 cos(theta)];

    Rphi = [ 1    0     0;
         0 cos(phi) -sin(phi);
         0 sin(phi) cos(phi)];

    Rib = Rpsi*Rtheta*Rphi;
```

```
    Q = [1     0     -sin(theta);
       0 cos(phi) sin(phi)*cos(theta);
       0 -sin(phi) cos(phi)*cos(theta)];

    M = [0   -w(3)  w(2);
       w(3)  0   -w(1);
      -w(2) w(1)  0  ];

    sys(1:3) = x(4:6);
    sys(4:6) = [0; 0; g] - Rib/m*[0; 0; ft];
    sys(7:9) = inv(Q)*w;
    sys(10:12) = J^-1*(tau-M*J*w);

  elseif flag == 3
    sys(1:12) = x(1:12);

  elseif flag == 4
    sampleTime = 0.01;
    sys = t + sampleTime;

  elseif flag == 0
    sizes = simsizes;
    sizes.NumContStates = 12;
    sizes.NumDiscStates = 0;
    sizes.NumOutputs = 12;
    sizes.NumInputs = 4;
    sizes.DirFeedthrough= 1;
    sizes.NumSampleTimes= 1;
    sys = simsizes(sizes);

    % initial conditions
    x0 = [i_x0(1) i_x0(2) i_x0(3) 0 0 0 0 0 0 0 0 0];
    str = [];
    ts = [ 0 0 ];
  else
    sys = [];
end
```

B. OFF-LINE RESULTS, LQT METHOD

The script in this section is used to create off-line LQT trajectories.

```
% Plot optimal intercept trajectory using LQT and straight line motion
% target trajectory.
% By Rob Allen
% --------------------------------------------------------------------------

clc
clear all
```

```
close all

% system parameters
A = [0 0 1 0;
   0 0 0 1;
   0 0 0 0;
   0 0 0 0];

B = [0 0;
   0 0;
   1 0;
   0 1];

H = [100000 0 0 0;
   0 100000 0 0;
   0 0 1 0;
   0 0 0 1];

Q = [1 0 0 0;
   0 1 0 0;
   0 0 1 0;
   0 0 0 1];

R = 100*[1 0;
   0 1];

tf = 35;
dt = 0.1;

x0 = [25; -100; 0; 0];
x_tf = [0; 0; 0; 0];

% Refrence trajectory
% 1: x=50 y=100 v=7 theta=-90 (Crossing)
% 2: x=100 y=25 v=7 theta=-165 (Head on)
% 3: x=-50 y=-25 v=7 theta=20 (Tail chase)
rv = 7;
% crossing
theta = -90;
r0 = [100; 50; rv*sind(theta); rv*cosd(theta)];

% Head on
% theta = -165;
% r0 = [25; 100; rv*sind(theta); rv*cosd(theta)];

% Tail chase
% theta = 20;
% r0 = [-25; -50; rv*sind(theta); rv*cosd(theta)];
I = 0;
for t=0:dt:tf
  I = I+1;
  time(I) = t;
```

```
  r_t(1:2,I) = r0(3:4)*t + r0(1:2);
  r_t(3:4,I) = [r0(3); r0(4)];
end

tmp1 = - B*inv(R)*B';
A_t = -A';
A_bar = ...
  [ A(1,1) A(1,2) A(1,3) A(1,4) tmp1(1,1) tmp1(1,2) tmp1(1,3) tmp1(1,4);
   A(2,1) A(2,2) A(2,3) A(2,4) tmp1(2,1) tmp1(2,2) tmp1(2,3) tmp1(2,4);
   A(3,1) A(3,2) A(3,3) A(3,4) tmp1(3,1) tmp1(3,2) tmp1(3,3) tmp1(3,4);
   A(4,1) A(4,2) A(4,3) A(4,4) tmp1(4,1) tmp1(4,2) tmp1(4,3) tmp1(4,4);
   -Q(1,1) -Q(1,2) -Q(1,3) -Q(1,4) A_t(1,1) A_t(1,2) A_t(1,3) A_t(1,4);
   -Q(2,1) -Q(2,2) -Q(2,3) -Q(2,4) A_t(2,1) A_t(2,2) A_t(2,3) A_t(2,4);
   -Q(3,1) -Q(3,2) -Q(3,3) -Q(3,4) A_t(3,1) A_t(3,2) A_t(3,3) A_t(3,4);
   -Q(4,1) -Q(4,2) -Q(4,3) -Q(4,4) A_t(4,1) A_t(4,2) A_t(4,3) A_t(4,4)];
eA_bartf =expm(A_bar*tf);

B_bar = dt*[ 0   0   0   0;
       0   0   0   0;
       0   0   0   0;
       0   0   0   0;
     Q(1,1) Q(1,2) Q(1,3) Q(1,4);
     Q(2,1) Q(2,2) Q(2,3) Q(2,4);
     Q(3,1) Q(3,2) Q(3,3) Q(3,4);
     Q(4,1) Q(4,2) Q(4,3) Q(4,4)];

% Solve for p(0) and x(tf)
tmp1 = 0;
I = 0;
for tau = 0:dt:tf
  I = I + 1;
  tmp1 = tmp1 + expm(A_bar*(tf-tau))*B_bar*r_t(:,I);
end

e1 = eA_bartf(1:8,1:4);
e2 = eA_bartf(1:8,5:8);
A(1:4,1:4) = eye(4);
A(5:8,1:4) = H;
A(1:8,5:8) = -e2;
B1 = e1*x0 + tmp1 + [zeros(4,4);H]*r_t(:,end);

X = inv(A)*B1;
xtf = X(1:4);
p0 = X(5:8);

% solve for the state and co-state
I = 0;
J = 0;
w0 = [x0; p0];
for t=0:dt:tf
  I = I+1;
  time(I) = t;
```

```
  % Integral term
  tmp1 = 0;
  II = 0;
  for tau = 0:dt:t
    II = II + 1;
    tmp1 = tmp1 + expm(A_bar*(t-tau))*B_bar*r_t(:,II);
  end
  w_t(:,I) = expm(A_bar*t)*w0 + tmp1;
  x = [ w_t(1,I); w_t(2,I); w_t(3,I); w_t(4,I)];
  p = [ w_t(5,I); w_t(6,I); w_t(7,I); w_t(8,I)];
  r = r_t(1:4,I);
  u(:,I) = - inv(R) * B' * p;
  J = J + ((x - r)' * Q * (x - r) + u(:,I)' * R * u(:,I)) * dt ;
end

% compute the optimal cost and velocity
J = 0.5*(x-r)'*H*(x-r) + 0.5*J
v = sqrt(w_t(3,:).^2 + w_t(4,:).^2);

% plot
axislimits = 120;
figure
plot(r_t(1,:), r_t(2,:), 'color', [0.8500 0.3250 0.0980], 'LineWidth', 2)
hold on

plot(w_t(1,:), w_t(2,:), 'color', [0.9290 0.6940 0.1250], 'LineWidth', 2)
hold off
xlabel('Y (m)')
ylabel('X (m)')
grid on
axis([-axislimits,axislimits,-axislimits,axislimits])
saveas(gcf,['output/H', num2str(theta),'_trajectory.eps'], 'epsc')

figure
subplot(2,1,1)
plot(time, u)
xlabel('time(s)')
ylabel('control input')
title(['Total cost: ', num2str(J)])
legend('F_x','F_y')

subplot(2,1,2)
plot(time, v)
xlabel('time(s)')
ylabel('velocity')
title('Velocity')
saveas(gcf,['output/H', num2str(theta),'_stats.eps'], 'epsc')
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] SkyTracker—Overview. (2017, May 9). CACI International Inc. [Online]. Available: http://www.caci.com/Skytracker/

[2] S. Hoogendoorn. (2017, Apr. 23) Guard from above. [Online]. Available: http://www.guardfromabove.com/

[3] C. F. Lin, *Modern Navigation, Guidance, and Control Processing*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1991.

[4] J. H. Blakelock, *Automatic Control of Aircraft and Missiles*, 2nd Ed., New York, NY: John Wiley & Sons, Inc., 1991.

[5] P. Zarchan, *Tactical and Strategic Missile Guidance*, 2nd Ed., Washington, DC: The American Institute of Aeronautics and Astronautics, Inc., 1994.

[6] A. Manjunath, "path following by a quadrotor using virtual target pursuit guidance," M.S. thesis, Dept. Mech. Eng., Utah State Univ., Logan, UT, 2016.

[7] R. Tan and M. Kumar, "Proportional navigation (PN) based tracking of ground targets by quadrotor UAVs," presented at the ASME 2013 Dynamic Systems and Control Conference, Palo Alto, CA, 2013.

[8] S. Park, J. Deysty, and J. P. How, "A new nonlinear guidance logic for trajectory tracking," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Providence, RI, 2004.

[9] E. Kahale, P. Castillo, and Y. Bestaoui, "Minimum time refrence trajectory generation for an autonomous quadrotor," in *International Conference on Unmanned Aircraft Systems*, Orlando, FL, 2014, pp. 126–133.

[10] Y. Bouktir, T. Haddad, and T. Chettibi, "Trajectory planning for a quadrotor helicopter," in *16th Mediterranean Conference on Control and Automation*, Ajaccio, France, 2008, pp. 1258–1263.

[11] M. Hehn and R. D'Andrea, "Real-time trajectory generation for interception maneuvers with quadrocopters," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal, 2012, pp. 4979–4984.

[12] T. Puls and A. Hein, "3D trajectory control for quadrocopter," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, 2010, pp. 640–645.

[13] I. Cowling, O. Yakimenko, J. Whidborne, and A. Cooke, “Direct method based control system for an autonomous quadrotor,” *J. Intell. Robot Syst.*, vol. 60, no. 2, pp. 285–361, Nov. 2010.

[14] N. S. Nise, *Control Systems Engineering*, 6th Ed., Hoboken, NJ: JohnWiley & Sons, Inc., 2011.

[15] J. Choon, “Development and validation of a controlled virtual environment for guidance, navigation and control of quadrotor UAV,” M.S. thesis, Dept. Mech. Eng., Naval Postgraduate School, Monterey, CA, 2013.

[16] I. D. Cowling, J. F. Whidborne, and A. K. Cooke, “Optimal trajectory planning and LQR control for a quadrotor UAV,” in *UKACC International Conference on Control*, Glasgow, UK, 2006.

[17] D. E. Kirk, *Optimal Control Theory, An Introduction*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1970.

[18] R. G. Hutchins, *Navigation, Missile and Avionics Systems*, Class Notes, [EC4330], Monterey, CA: Naval Postgraduate School, 2005.

[19] L. R. G. Carrillo, A. E. E. Lopez, R. Lozano, and C. Pegard, *Quad Rotorcraft Control, Vision-Based Hovering and Navigation*, London, UK: Springer-Verlag, 2013.

[20] E. F. Camacho and C. Bordons, *Model Predictive Control*, London, UK: Springer-Verlag, 1999.

[21] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design*, Madison, WI: Nob Hill Publishing, LLC, 2009.

[22] MathWorks, Inc. (2017, May 3). Model predictive control toolbox. [Online]. Available: https://www.mathworks.com/help/mpc/index.html

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California